



**PARAMETER SETTING SOFTWARE
LT160
MODULAR INPUTS OUTPUTS**

OPAL

USER MANUAL



FOREWORD

This manual provides the information required to run LEROY Automatique Industrielle LT's.

The software implementation is realised with the **OPAL software (LT Parameter setting tool)**.

Computer programmers and control engineers will find the necessary information concerning the procedures for implementing the OPAL parameter setting tool on the PC, the parameterisation of the functions, the data processes and manipulation, the memory mapping definitions available over the network and led diagnostics.

The equipment implementation is presented in the “LT Implementation Manual ».

NB : the LT160 must have the embedded OPAL destined software.

Technical support :

Tel : (33).05.62.24.05.46
Fax : (33).05.62.24.05.55
e-mail : support@leroy-autom.com

LEROY Automatique Industrielle regularly develops and improve their products. The information contained within this documentation is liable to modification without prior warning and under no circumstances does it represent a commitment on behalf of the company. This manual may not be duplicated in any form without the approval of LEROY Automatique Industrielle.

Leroy Automatique Industrielle

Head-office : Boulevard du Libre échange
31650 Saint Orens
Tel : (33).05.62.24.05.50
Fax : (33).05.62.24.05.55

CONTENTS

I- IMPLEMENTATION OF THE LT PARAMETER SETTING TOOL: OPAL	1
I.1- REQUIRED EQUIPMENT AND SOFTWARE	1
I.2- OPAL INSTALLATION AND OPERATION UNDER WINDOWS	1
I.2.1- <i>Installing OPAL under Windows</i>	1
I.2.2- <i>Launching OPAL under Windows</i>	2
I.3- MAIN MENU	2
I.4- OPAL TO LT CONNECTION: SWITCHING TO PARAMETER SETTING MODE (PRM)	3
I.4.1- <i>Parameter setting mode on the LT</i>	3
I.4.2- <i>Parameter setting mode on the LT80 and LT160</i>	4
II- OPAL QUICK START	5
III- EQUIPMENT PAGE: SETTING THE LT EQUIPMENT CONFIGURATION PARAMETERS	9
IV- PARAMETERS PAGE.....	11
IV.1- GENERAL	11
IV.2- SETTING THE PARAMETERS OF THE POWER SUPPLY BLOCK (LT100)	12
IV.2.1- <i>Alarms and Diagnostics</i>	12
IV.3- PARAMETER SETTING OF THE CENTRAL UNIT BLOCK.....	12
IV.3.1- <i>Clock</i>	12
IV.3.2- <i>Alarms /Communication synthesis (except Ethernet channel)</i>	13
IV.3.3- <i>Alarms synthesis / Wiring control</i>	13
IV.3.4- <i>Diagnostics / central unit status word</i>	14
IV.3.5- <i>Diagnostics / Central unit cycle time</i>	14
IV.3.6- <i>Diagnostics / Excess cycle time alarm</i>	15
IV.4- SETTING THE PARAMETERS OF A JBUS/MODBUS SLAVE CHANNEL.....	15
IV.4.1- <i>Transmission parameters</i>	15
IV.4.2- <i>Additional silence time</i>	15
IV.4.3- <i>Communication alarm</i>	15
IV.4.4- <i>Diagnostics / Counters</i>	16
IV.5- SETTING THE PARAMETERS OF A JBUS/MODBUS MASTER CHANNEL	16
IV.5.1- <i>Transmission parameters</i>	16
IV.5.2- <i>Principle of frame transmissions</i>	16
IV.5.3- <i>Communication alarms</i>	17
IV.5.4- <i>Diagnostics / Counters</i>	17
IV.6- SETTING THE FIP CHANNEL PARAMETERS (ON THE LT100 AND LT80).....	17
IV.7- SETTING THE ETHERNET CHANNEL PARAMETERS (ON THE LT100 AND LT80).....	17
IV.7- SETTING BINARY INPUT PARAMETERS (TOR).....	19
IV.7.1- <i>Direct inputs</i>	19
IV.7.2- <i>Filters</i>	20
IV.7.3- <i>Timers</i>	20
IV.7.4- <i>Edge Counters</i>	20
IV.7.5- <i>Wiring control (DI312 modules solely)</i>	21
IV.8- DIGITAL OUTPUT PARAMETER SETTINGS	22
IV.8.1- <i>Command of digital outputs</i>	22

IV.8.2- Output signal processes.....	23
IV.8.3- Processes / Fallbacks.....	25
IV.9- SETTING PARAMETERS OF ANALOG INPUTS	26
IV.9.1- Direct analog inputs.....	26
IV.9.2- Filters.....	27
IV.9.3- Scaling.....	27
IV.9.4- Thresholds.....	28
IV.10- SETTING ANALOG OUTPUT PARAMETERS	29
IV.10.1- Analog output command.....	29
IV.10.2- Scaling.....	29
IV.10.3- Filters.....	30
IV.10.4- Fallback.....	30
V- DATA PAGE	31
V.1- THE PUBLIC DATA LISTS	31
V.2- LIST OF ALL THE LT DATA	31
V.3- OPERATIONS ON THE DATA LISTS	33
V.4- DATA ACCESS TABLE VIA TERMINAL BLOCK X CHANNEL Y (MODBUS/JBUS SLAVE)	34
V.5- DATA ACCESS TABLE VIA TERMINAL BLOCK X CHANNEL Y (MODBUS/JBUS MASTER).....	34
V.5.1- Adding or modifying a frame	35
V.5.2- Implemented functions.....	35
V.5.3- Conditional triggering and exchange reports.....	36
V.5.4- Frame sequencing in the LT cycle	37
V.6- DATA ACCESS TABLE VIA TERMINAL BLOCK 1 CHANNEL 0 (FIP)	38
V.6.1- Parameter setting synthesis for the implementation of FIP variables.....	38
V.6.2- Variable identifiers.....	38
V.6.3- Variable types.....	39
V.6.4- Variable production and consumption	40
V.6.5- Variable refresh status.....	40
V.6.6- Variable promptitude status	41
V.7- DATA ACCESS TABLE VIA TERMINAL BLOCK 1 CHANNEL 0 (OPENMODBUS/TCP SLAVE)	42
V.8- DATA ACCESS TABLE VIA TERMINAL BLOCK 1 CHANNEL 0 (SNMP)	43
VI- LT CONNECTED PAGE	45
VI.1- WRITE CONFIGURATION.....	45
VI.2- SIZE OF THE ENTERED CONFIGURATION	45
VI.3- REFRESH.....	45
VI.4- DYNAMIC INPUT/OUTPUT BLOCKS.....	45
VI.5- COMMUNICATION PARAMETERS	46
VI.6- JBUS COMMANDS	46
VI.7- JBUS COMMAND REPORTS	47
VII- PROCESSES PAGE	49
VII.1- REMINDER : DATA DEFINITION	49
VII.1.1- Data types	49
VII.1.2- Internal data addresses : Alignment	50
VII.2. WRITING A PROCESS	50
VII.2.1- Expression comment and terminator	50
VII.2.2- The operators.....	50
VII.2.3- Typecasting	52
VII.2.4- Control instruction.....	53
VII.2.5- Conditional loop	53
VII.3- THE FUNCTIONS.....	55
VII.3.1- General syntax	55
VII.3.2- List of available functions	56
VII.3.3- Edge detection of source bits	56
VII.3.4- Boolean synthesis with authorisations (Mask).....	58
VII.3.5- Digital functions.....	60
VII.3.6- Timer functions	64
VII.3.7- Edge filters.....	67
VII.3.8- Square and rectangular signals	68
VII.3.9- Monostable.....	68
VII.3.10- Impulse.....	69

VII.3.11- Comparisons of double words (comparing dates and times).....	70
VII.4- THE PROCEDURES	72
VII.4.1- General syntax.....	72
VII.4.2- List of available procedures	73
VII.4.3- Transfer procedure on state of condition	74
VII.4.4- Transfer procedures on the rising edge of a condition.....	76
VII.4.5- Transfer procedures on the rising and falling edge of a condition	77
VIII- DIAGNOSTICS AND IMPLEMENTATION	79
VIII.1- DIAGNOSTICS OF THE PSD230 (LT100) POWER SUPPLY	79
VIII.2- DIAGNOSTICS OF THE CENTRAL UNITS CPU1XX (LT100).....	79
VIII.3- DIAGNOSTICS OF THE PSD300 (LT80) - PSD301 (LT160) POWER SUPPLIES	80
VIII.4- DIAGNOSTICS OF THE CPU3XX CENTRAL UNITS (LT80 AND LT160)	80
VIII.5- DIAGNOSTICS OF THE COMXXX COMMUNICATION MODULES	81
VIII.6- DIAGNOSTICS DIXXX DIGITAL INPUT MODULES	81
VIII.7- DIAGNOSTICS OF THE DOXXX DIGITAL OUTPUT MODULES.....	81
VIII.8- DIAGNOSTICS OF THE DIO210 DIGITAL INPUTS/OUTPUTS MODULES.....	81
VIII.9- DIAGNOSTICS OF THE AIXXX ANALOG INPUT MODULES	82
VIII.10- DIAGNOSTICS OF THE AOXXX ANALOG OUTPUT MODULES	82
VIII.11- DIAGNOSTICS OF THE AIO320 ANALOG INPUT/OUTPUT MODULES.....	82
FIGURE TABLE	83

I- Implementation of the LT Parameter setting Tool: OPAL

I.1- Required equipment and software

Equipment :

- Minimum configuration: A PC compatible computer (4 Mb min.) with a serial port for the LT connection,
- 2 Mb of available hard drive space for the OPAL directory installation,
- a RS232 LT-PC cable wired as below :

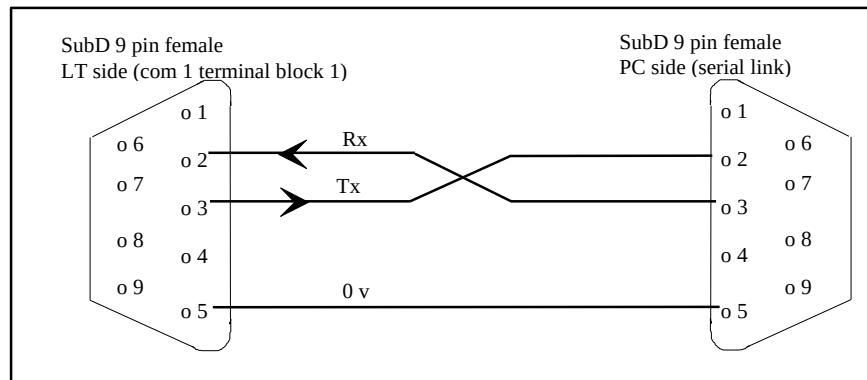


Figure 1 : LT ↔ PC connection wiring

Software :

- Microsoft Windows XP,
- The OPAL diskette.

I.2- OPAL installation and operation under Windows

I.2.1- Installing OPAL under Windows

- Insert the OPAL diskette in the disk drive, which we assume is A:\
- Execute the command "A:\install.exe"
- Enter your company name, user name, and installation directory (by default C:\OPAL)
 - ◊ Use the tab key on the keyboard to pass from one field to another.
 - ◊ Activate the "Install" button
 - ◊ The installation window opens
 - ◊ Files are installed
 - ◊ A dialog window opens informing you that "OPAL was installed at xxxxxx"
 - ◊ Validate with OK. This is followed by a dialog window " Create program groups " : Yes/No/Cancel
- If YES , Opal will create the **OPAL group** and display the **OPAL and Notes icons**. Activating the "Notes" icon opens the ReadMe.txt. This contains all the information that may not yet be documented. Activating the "OPAL" icon launches OPAL.
- NO , terminates and exits the installation procedure.

I.2.2- Launching OPAL under Windows

- For versions of Windows 3.1x
 - ◊ Open the OPAL group
 - ◊ Activate the OPAL icon
- For the 9x and NT4 versions
 - ◊ Open the **Startup** menu
 - ◊ Select the **Programmes** sub-menu
 - ◊ Select the **OPAL** sub-menu (the program group)
 - ◊ Select and activate the OPAL sub-menu.

The main OPAL presentation window will appear displaying :

- The OPAL version number,
- The OPAL version date,
- The company name,
- The user name.

Note : The DTR is active while OPAL is running.

Press OK. OPAL will immediately try to detect an LT presence. If an LT is not present or unrecognisable an empty "OPAL - EQUIPMENT" configuration window will be opened. If an LT has been recognised, OPAL will open the "LT connected" window.

I.3- Main Menu

The OPAL main menu displays the following menu headers : **File** **Pages** **(PageName)** **Help**

Note : The **underlined character** enables menu selection via keyboard shortcuts in the ALT + "underlined character" form.

(PageName) is the name of the active page and is dependent on the context. OPAL consists of the following pages : Equipment , Parameters, Data , Process , LT connected. When a page is selected, the main menu is refreshed to indicate the name of the corresponding page.

File is a standard application menu under Windows. It allows you to manipulate the configuration file. OPAL attributes the .OPA suffix by default and displays the list of .OPA files.

Pages enables you to pass from one page to another. The other possibilities of passing from page to page are :

- Using the << and >> buttons at the bottom left and right of each page
- Using the PageUp and PageDown keys of the keyboard.

LT configuration requires :

- Completion of the 4 successive "**E**quipment", "**P**arameters", "**D**ata" and "**P**rocess" pages. Each page uses information from the preceding page.
- The saving of the configuration to a file.
- The connection of the PC to the LT and passing via the "LT connected" page in order to "Write the configuration" .

WARNING : the configuration parameters generated by OPAL cannot be retrieved from the LT. Only the equipment status can be updated dynamically. **Keep your configuration file (.OPA file).**

I.4- OPAL to LT connection: switching to parameter setting mode (PRM)

The LT should be in parameter setting mode in order to ensure that OPAL can read or write a configuration. This mode sets the Com1 communication parameters to the default values : Jbus slave protocol, slave number 1, 19200 bits/s, 8 data bits, no parity, 1 stop bit).

Important : In PRM mode the processes and communications of the other ports are no longer executed. The auto-test is however completely executed.

I.4.1- Parameter setting mode on the LT

To get into PRM :

- Connect the LT (terminal block 1, Com1) and the PC (Com1 or Com2) with a RS232 cable ,
- Unplug the LT100,
- Bridge the **2 PRM points** of the power connector,
- Plug in the LT100,
- Remove the PRM bridge,
- Set up the LT100 parameters under OPAL (refer to following chapters),
- Connect OPAL to Com1 of terminal block 1. Go to the OPAL "LT connected" page,
- Request configuration write.

Note : Once the new configuration has been loaded, OPAL orders the LT100 re-initialisation. If the PRM bridge has not been removed the LT100 will restart with the default configuration : remove the bridge and switch back on .

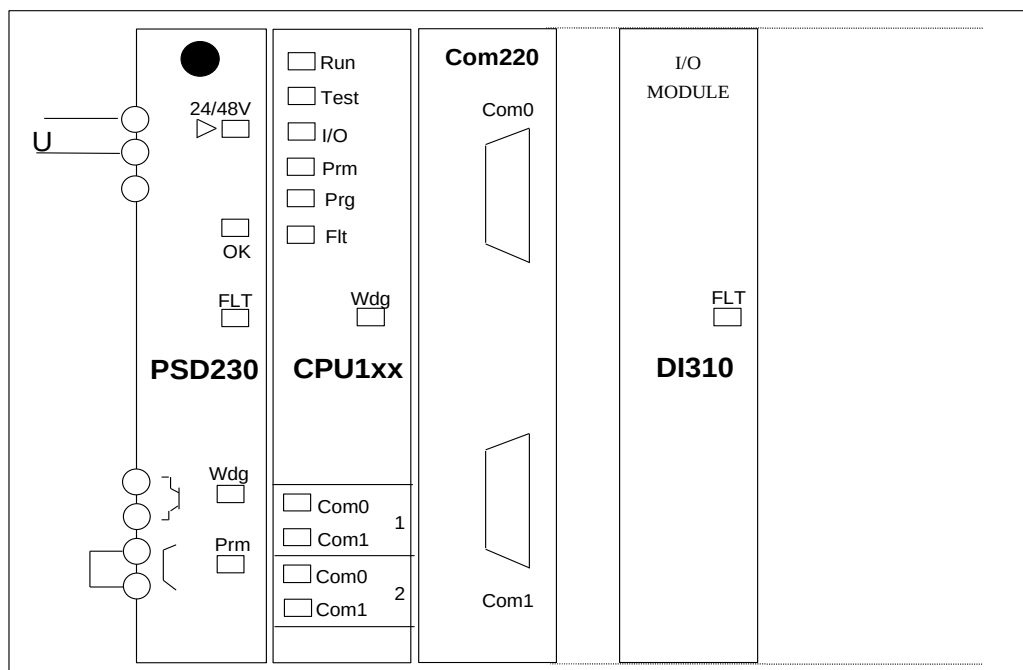


Figure 2 : LT100 PRM mode wiring

Expected results after PRM mode startup :

- The green 24/48V LEDs and OK are permanently lit,
- The orange power PRM LED is permanently lit,
- The green Run LED blinks rapidly,
- The orange CPU Prm LED is permanently lit. The procedures are not executed (refer to following chapters).
- The green Com1 LED on the terminal block 1 is permanently lit,
- The WDG, PRG, and FLT supply LEDs should not be lit,
- The status of other LEDs are unimportant.

I.4.2- Parameter setting mode on the LT80 and LT160

Switching to PRM from MSDOS :

- Switch the LT off,
- Connect the LT (Terminal block 1, com1) and the PC (com1 or com2) with a RS232 cable (figure 1),
- Execute the "LTPRM.EXE" program, supplied on the " OPAL " diskette, under DOS. This program will expect to recognise an LT. It will display "**Waiting for connection on com 1**". The wait can be terminated by pressing a key on the keyboard,
- Switch the LT on,
- During the initialisation sequence the LT will switch to PRM and light up the PRM led. The PC will display "**Success, PRM mode command sent.**"
- The LT is now in PRM mode
- Set the LT80 and LT160 parameters under OPAL (refer to following chapters),
- Connect OPAL to Com1 of the terminal block 1. Go to the OPAL "LT connected" page,
- Request configuration write.

Switching to PRM under Windows95 :

- Switch the LT off,
- Connect the LT (terminal block 1, com1) and the PC (com1 or com2) with a RS232 cable (figure 1),
- Execute the "LTPRM.EXE" program, supplied on the " OPAL " diskette, under Windows95. This program will expect to recognise an LT.
- Select "**Startup**",
- It will display "**Exceeding idle time. Waiting for acknowledgement character**"
- The wait can be terminated by pressing "**Stop**",
- Switch the LT on,
During the initialisation sequence the LT will switch to PRM and light up the PRM led. The PC will display "**LT parameter settings by default**"
- The LT is now in PRM. Mode,
- Set the LT80 and LT160 parameters under OPAL (refer to following chapters),
- Connect OPAL to Com1 of the terminal block 1. Go to the OPAL "LT connected" page,
- Request configuration write.

Important : only the permanent lighting of the PRM led guarantees a successful switching to PRM mode.
No specific cabling is required to switch to PRM mode on the LT80 and LT160.

Expected results after PRM mode startup :

- The green OK LED is permanently lit,
- The green PRM LED is permanently lit,
- The green Run LED blinks rapidly,
- The green Com1 LED on terminal block 1 is permanently lit,
- The WDG LED should not be lit,
- The status of other LEDs are unimportant.

II- OPAL Quick Start

In this chapter we will set the parameters of an **LT80 with 3 inputs/outputs boards**. The setting of these parameters will give us an overview of LT parameter settings without having to go into detail.

The LT is equipped with the following :

- PSD300
- CPU301
- DI310
- DO310
- AI210

On the Com1 (Jbus slave) we will show how a master Jbus/Modbus can access the information on an LT. OPAL supplies a "Master Jbus/Modbus" tool in the "LT connected" page

On the Com0 (Jbus Master) we will show how to access the information of a LEROY Automatique Industrielle TES 32ST (Jbus Slave, 32 digital outputs).

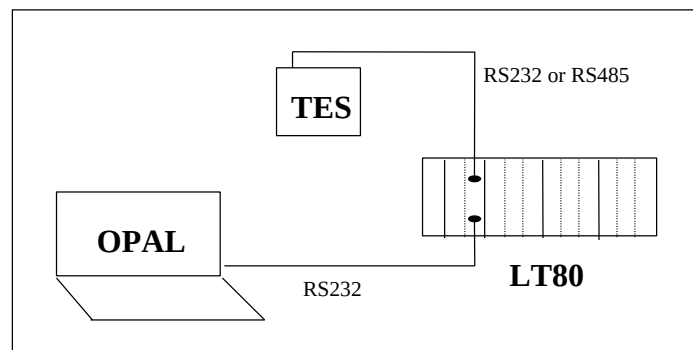


Figure 3 : The example network architecture

This example will be achieved in the **5 steps**, which correspond to the **5 OPAL pages** :

Launch OPAL

If the LT is not connected to Com1 of the PC then OPAL will display the Equipment page immediately. Otherwise, use the menu or arrows at the bottom left or right.

Step 1- Equipment page

Choose the following by double clicking on each field :

- Power supply block : PSD300
- Central Unit : CPU301
- Communication Terminal Block : Com301
- Inputs-outputs blocks
 - ◊ 1 : DI310
 - ◊ 2 : DO310
 - ◊ 3 : AI210

Step 2- Page Parameters

Double click "Terminal block 1 channel 0 : -" in the "Communication channels" part. Select the Jbus Master protocol. Set the speed to 9600 bauds and the parity to even in the default parameter window. These parameters are identical to the TES default parameters.

All of the fields on the "Parameters" page have a « **Default parameters** » comment apart from the "Communication channels" field which now has a "**Customised parameters**" comment.

Step 3- Data Page

By default this page displays the "LT list of public data ". The list contains :

Name	Type	Len	Description
bi311001	BIT	32	Block 1 : Digital inputs
bo411002	BIT	32	Block 2 : Real digital outputs
bo412002	BIT	32	Block 2 : Digital outputs command
mi512003	WORD	16	Block 3 : ANA Inputs

Each variable name may be used as in the process page or may be renamed by using the "Modify" button.

Entering a Jbus Master frame: select " Terminal block 1 channel 0 (Jbus master) data access table" (highlighted in the main menu). The window's "Add" button opens the parameter setting window for a Jbus Master frame. Choose :

- slave : 1
- function : write bits
- Jbus address: 80
- Jbus length : 32
- variable to exchange : bo411002 (Block 2 : Real digital outputs)

This frame will be executed cyclically by the LT : as soon as a frame is finished (slave reply or time-out) the next is launched. This enables the copying the status of the 32 digital outputs of the LT80 DO310 board to the digital outputs command of the TES 32ST.

Data access by the Jbus slave link : select the "Com 1 data access table" (Jbus Slave) highlighted in the main menu. This list contains :

Address	Type	Length	Description
0000	BIT	00032	Block 1 : Digital inputs
0020	BIT	00032	Block 2 : Real digital outputs
0040	BIT	00032	Block 2 : digital outputs command
0006	WORD	00016	Block 3 : ANA inputs

For a Jbus/Modbus master:

- The "Digital inputs" can be accessed at the **word 0 address** or **bit 0 address**
- The "Real digital outputs" can be accessed at the **word 2 address** or **bit 20 (hex) address**
- The "Digital outputs command" can be accessed at the **word 4 address** or **bit 40 (hex) address**
- The "ANA inputs " can be accessed at the **word 6 address** for input 1, at the **word 7 address** for input 2 and so on for the 16 inputs.

Step 4- Processes Page

A process example, which uses a procedure, supplied by OPAL : copy the 16 analog inputs into a 16 word data if the digital input corresponds to 1.

Declare a 16 word data beforehand in the data page: *EANAAutorisees*;
`_MOV(mi511003, bi311001[0..15], EANAAutorisees);`

Step 5- LT connected Page

Com1 of the LT80 terminal block 1 should be connected to the PC Com1. This LT port is configured with the same parameters as the OPAL (19200 baud, no parity , 1 stop, 8 data bits) by default.

Click the "**Write Configuration...**" button

OPAL displays a window, which indicates the available space on the LT80 as well as the space taken by your configuration. Click "SEND".

OPAL transfers the configuration to the LT. At the end of the transfer OPAL gives the **LT80 a re-initialisation command**. As soon as the Jbus exchanges are re-established the **"LT connected" page is refreshed once**. Each LT element can be dynamically diagnosed with a double click on the desired element..

The "Refresh" button enables the reading of the equipment configuration and the elements associated with the "LT connected" page.

The "**Jbus commands...**" button opens a window which enables the entry and launching of Jbus Master frames. Only one frame can be entered and launched at a time. An example of frames to enter :

- Read the digital inputs on the LT80: LMOT 1 0 2 then "Execute".
- Read the ANA inputs on the LT80 : LMOT 1 6 16 then "Execute".

This tool may also be used for any other Jbus/Modbus slave. For instance the TES.

III- EQUIPMENT PAGE: setting the LT equipment configuration parameters

The **Equipment** page allows you to enter your LT equipment configuration. The LT100 and LT160 allow you to insert a **maximum of 15 blocks** numbered from 1 to 15. Only **3 output/input modules** can be configured for the **LT80**. The blocks are numbered from left to right. The PSDxxx power supply and the CPUxxx central unit and the COMxxx communication terminals blocks constitute block 0. This block is called the "**line head**". The numbering continues from left to right if your LT has an extension base. The **EXT100 and EXT101** extension couplers and the **terminal cabling block modules** do not take a number as they are completely transparent to OPAL and are therefore not recognised.

Enter the corresponding module serigraph reference in each entry field..

Activation of the field opens the list of available modules that can be selected for this field.

To activate a field :

- possibility 1 : double click the desired field. Use the scroll bar if necessary
- possibility 2 : Open the Equipment menu in the main menu

Selecting **Clear Configuration** in the **Equipment** menu clears all previously chosen modules.

In order to **place** a module, choose the desired reference in the list. To **remove** a module, choose it in the first choice list (-).

Remark about the communication terminal blocks : they are functionally linked to the central unit. The choice of the latter infers settable terminal blocks. Field 1 corresponds to the closest CPU module placement (terminal block 1). The protocols associated with each communication port are entered in the "Parameters" page.

The following table shows the different equipment available for the LT100, LT80 and LT160 line head:

	LT100	LT80	LT160
Power supply	PSD230	PSD300	PSD331
Central units	CPU100 CPU110 CPU112 CPU120 CPU130 CPU132	CPU301 CPU302 CPU303	CPU332 CPU334 CPU354
Inputs/outputs blocks	DI310 DI410 DO310 AI110 AO120 AI220 AO220 DI312	DI310 DI410 DO310 AI110 AO120 AI220 AO220 DI312 DIO210 AIO320 NC100	DI310 DI410 DO310 AI110 AO120 AI220 AO220 DI312 DIO210 AIO320 NC100

IV- PARAMETERS PAGE

IV.1- General

The functional module parameter setting consists of :

- setting certain parameter values,
- render the internal information on the LT accessible to the operating network.

The Parameters page consists of 4 parts. The fields should be filled in order to set the equipment that was selected on the previous "Equipment page" :

POWER SUPPLY (on the LT100) : activating this field opens the settings window for this power supply.

CENTRAL UNIT : activating this field opens the settings window for this central unit.

COMMUNICATION CHANNELS: one or more communication channels are associated with each CPU type. Depending on the LT type, there are 1 to 4 terminal blocks marked respectively "**terminal block 1**" to "**terminal block 4**", going from the left to the right of the LT. The upper communication channel on each terminal block is called "**channel 0**" and the lower is called "**channel 1**".

Here are the different terminal blocks that are available according to the LT type :

LT100	LT80	LT160	Comments
COM020			1 RS232/485 channel
COM220			2 RS232/485 channels
COMF20			1 RS232/485 channel + 1 FIP channel
	COM301	COM301	1 RS232/prg channel + 1 RS232/485 channel
	COM302		1 RS232/prg channel + 1 FIP channel
		COM311	2 RS232C channels
		COM312	2 RS232/485 channels
	COM303	COM303	1 RS232/prg channel + 1 Ethernet channel

The activation of a field displays a list of available protocols for each communication channel and then opens the protocol parameter setting window (refer to following chapter)

INPUTS - OUTPUTS : this part offers a choice by **DATA TYPE** (Digital inputs, Digital outputs, ANA inputs, ANA outputs) according to the presence of the declared modules. Field activation opens the function parameter settings window for the data type.

The parameters that are set in the Parameter page automatically create the operator useable data and are listed in the lists available in the Data page..

Selecting **Default setting** in the **Parameters** menu clears all previously defined parameters.

IV.2- Setting the parameters of the power supply block (LT100)

The reference to the chosen power supply in the Equipment page is remembered. Those functions unavailable to the block appear greyed out.

IV.2.1- Alarms and Diagnostics

The power supply contains the status, alarms and diagnosis bits. The explanation of which is given below.

- **Parameters Page:** check the checkbox to authorise network read access of this status word. It will become available in the internal data list.
- **Data Page:** adds a 16 bit data called "Power supply diagnostics".

Only bits 4 and 7 are used :

Order	Bit signification
4	PRM input bit status (bit at 0 if PRM bridge present)
7	"Wdg Pow" bit status (see below)

The WatchDog output (WDG) :

The watchdog is active (ON) if

- Defective internal power
- Or if faulty internal sensor
- Or if faulty central unit
- Or if ordered by the central unit (via the network)

Wdg Pow	Terminal block negative Boolean	LED	Status of bit 7
inactive (OFF)	Outgoing current : 100mA (24V)	off	1
active (ON)	Outgoing current: nil	lit	0

IV.3- Parameter setting of the central unit block

This page allows you to set the parameters of the central units. The reference to the central unit chosen in the Equipment page is remembered. Functions that are not available to the chosen block will appear greyed out. This page proposes the parameter settings for the following functions :

IV.3.1- Clock

The clock function is a software clock, which can be set to the equipment clock, if the central unit is fitted with one. Only those central units fitted with the HTR component ensure off line clock management.

Warning : hundredths of seconds are not battery protected.

Parameters Page: activation of the clock function (checkbox: "Put the clock in the data list ") reserves 4 double words in the data list : the first quadruple word is the current time stamp, the second is the time setting value written by the network master. The time stamp format is described below. The software clock compensation drift comprises of adding or subtracting 1 ms every N seconds.

Data Page : adds the current time stamp quadruple word and the quadruple time reset word. Adds a 4 word data called *clock* :

Order		High order byte	Low order byte
0	Current time stamp	seconds	1/100ths of a sec.
1		Hours	minutes
2		Day of month	Day of week (**)
3		Year	Month
4	Time reset time stamp	seconds [0..59]	00 (*)
5		hours [0..23]	minutes [0..59]
6		Day of month[1..31]	Day of week(**) [0..6]
7		year [0..255]	month [1..12]

(*) the hundredths of seconds must be nil when resetting the time stamp.

(**) The user initialises the day of the week, between 0 and 6. The 0 does not necessarily have to be Monday. This byte is incremented each day. If the byte is initialised with a value greater than 6 it will maintain that value until a change of day and will be reset to 0..

Time reset mechanism via the network : The time update values are initialised at FFh. If 0 is written to the byte for the hundredths of a second time stamp of the time reset (WORD 4 : low order) then the LT will copy the time stamp time reset to the current time stamp and reset the update time stamp to 0xFFFFFFFF.

Example : Passage to the year 2000 : time reset on Friday the 31st of December 1999 at 23h55mn00s and time reading 1 hour later. The chosen year of reference is 99 and the day of the week is 4. The LT is configured with a Jbus slave channel with a slave number of 1 ; the clock has a network address of 0..

EMOT 1 4 4 0000 1737 1F04 630C ⇒ Ok

Time stamp reading on Saturday the 1st of January 2000 at 0h55mn00s :

LMOT 1 0 8 ⇒ 00xx* 0037 0105 6401 FFFF FFFF FFFF FFFF

* the hundredths number depends on the moment of reading

NB : for the LT80 and LT160, an incorrect time update will stop the software clock until the correct time has been set. The time update can also be partial : the fields to FFh retain their current value.

IV.3.2- Alarms /Communication synthesis (except Ethernet channel)

Each communication channel associated with a CPU can generate a communication alarm (refer to setting the parameters of the communication channels). The channel alarm bit is :

- Set to 1 if no communication
- Reset to 0 with renewed communication

The central unit can achieve the synthesis, with AND (**all channels on alarm**) or the synthesis with OR (**at least one channel on alarm**), of the communication faults of the channels.

Parameters Page : insert the channel alarm in the general alarm. In order to do this the channel must generate an alarm : Parameters Page/Communication channels. Choose the combining synthesis equation for the central unit alarm : AND synthesis, OR synthesis. Buttons are not available if no channel is selected. If only one channel is selected then the OR synthesis will be available.

Data Page: adds a data bit labelled *Synthesis of communication alarms*

IV.3.3- Alarms synthesis / Wiring control

This function enables the generation of :

- An OR synthesis bit of the wiring control for all the inputs of the same wiring control block. (block X : Wiring alarm)

- An OR synthesis bit of the Wiring alarm bits for all the wiring control blocks (Wiring alarms synthesis)

Parameters Page: indicate those boards participating in the synthesis on the list of available boards.

Data Page: add 1 bit synthesis data labelled *Wiring alarms synthesis* per security inputs block. Adds a 1 bit general synthesis data labelled *Wiring alarms*.

IV.3.4- Diagnostics / central unit status word

Parameters Page: check the checkbox to authorise network read access for this status word by placing it in the internal data list.

Data Page: addition of a 16 bit data labelled *Central unit status word* :

Order	Signification	The bit is 1 :
0	End of Cycle	At the end of each cycle
1	Wdg status	If the CU Wdg LED is lit
2	I/O defect	If defect on internal bus
3	Init end	If initialisation has ended = Cycle running (*)
4	Auto-test error	If at least one error has been detected by the auto-test
5	PRM state	if a PRM state has been detected
6	Int 0	To 1 if processor calculation error
7	reserved	reserved
8 to F	reserved	reserved (**)

(*) Bit forced at each cycle (to 0 during initialisation, to 1 once initialisation ended).

(**) This byte holds the CTS state (1 or 0) on the LT100.

Central unit watchdog : the " CU Wdg " is active (bit 1 of CPU status at 1, CPU led lit) if :

- the microprocessor is on reset (Initialisation or breakdown)
- there is no internal power
- or if there is a defective power sensor
- or if the CU is defective

On the LT80 and LT160 :

Wdg Pow	Relay output on terminal block	Wdg Led
inactive (OFF)	Open contact	off
active (ON)	Closed contact	on
LT power off	Closed contact	off

IV.3.5- Diagnostics / Central unit cycle time

The central unit measures its cycle times and gives the user three measures.

Parameters Page : check the checkbox to authorise network read access of the cycle times by placing them on the internal data list.

Data Page : adds a 3 word data labelled *Central unit performance times*:

Order	Signification
0	Duration of last completed cycle
1	Maximum recorded duration
2	Minimum recorded duration

IV.3.6- Diagnostics / Excess cycle time alarm

The LT cycle consists of the inputs acquisition, processing network access, digital processing and outputs writing. The duration of the cycle is measured (see above) and compared to a maximum authorised duration parameter. An alarm is set to 1 if the cycle exceeds the maximum authorised duration.

Parameters Page : authorise network read access of this alarm bit by placing in on the internal data list. Enter the maximum cycle duration in ms.

Data Page : adds a 1 bit data labelled *Excess cycle duration alarm*.

IV.4- Setting the parameters of a Jbus/Modbus slave channel

The accessible data through this channel is available in the Data Page. The Jbus/Modbus addresses for this data can be seen by displaying *The data access tables via "Terminal blockx " Channely (protocol)*.

IV.4.1- Transmission parameters

The number of data bits is set to 8. The slave number must be between 1 and 255. The other parameters are :

- speed : 150 to 38400 bits/s,
- parity : even, odd, none,
- stop bits: 1, 2.

IV.4.2- Additional silence time

This period of additional **inter-frame** silence is added to the time that the slave takes to process a query. This enables a "wait" before sending the reply. The time is entered in milliseconds.

IV.4.3- Communication alarm

Alarm type : a slave Jbus channel can generate a communication alarm :

- if it no longer detects queries from the master (even if they are not addressed to it) or,
- if it doesn't detect any activity on the network.

This alarm participates in the central unit general communication alarm synthesis. The corresponding LED of the communication channel enables alarm visualisation on the CPU (refer to Diagnostics chapter).

Triggering the alarm : the defect should be confirmed during at least N ms before generating the alarm bit N comprised between 0 and 9999 ms.

Alarm authorisation : the LT can initiate line monitoring as from initialisation or as from the moment when it first detects line activity.

Parameters Page: set the alarm type, the confirmation prior to alarm duration, alarm triggering.

Data Page : adds a 1 bit data labelled *Channel alarm bit Terminal block x Channel y (protocol)*.

IV.4.4- Diagnostics / Counters

Parameters Page : Check the checkbox to put the diagnostics counters in the internal data list.

Data Page : adds a 9 word data labelled *Frame counters channel Terminal block x channel y (protocol)*

Order	Signification
0	Number of frames received whether the LT was concerned or not
1	Number of frames received with CRC16 errors
2	Number of exception replies sent by the LT
3	Number of correct frames received and executed by the LT
4	Number of broadcast frames received by the LT
5	unused
6	Number of "Slave not ready" replies returned by the LT
7	Number of erroneous characters received by the LT
8	Number of incoherent frames received by the LT

IV.5- Setting the parameters of a Jbus/Modbus master channel

The frames to be transmitted by the LT and the trigger conditions are to be defined in the Data Page that corresponds to this channel : *Data access table via terminal block x channel y (protocol)*. We will just be describing the parameter settings for a Jbus/Modbus master channel in this paragraph.

IV.5.1- Transmission parameters

The number of data bits is set to 8. The other parameters are :

- **speed** : 150 to 38400 bits/s,
- **parity** : even, odd, none,
- **stop bits** : 1, 2.

IV.5.2- Principle of frame transmissions

The network master sends a request (or query) to a slave. The complete reply, from the slave, must have arrived before the **Maximum duration for a complete exchange** in (milliseconds) has been reached. The period starts as from the transmission of the first character by the master.

If the reply has not returned the master will resend the command until it attains **the default number of transmission attempts**. This is normally set to 3.

The master could have received the "Slave not ready" exception reply. An alternative resend number parameter is available for this case : **Number of slave not ready attempts**.

The master decides a reply from the slave has been received if the line silence period after the last character received is equal to the duration of 3 characters (duration varies according to the speed). This duration may occasionally be insufficient especially when using modems. An **Additional inter-Frame silence period** (in milliseconds) may therefore be added to these three characters. Please note that the duration is added in milliseconds and not in characters. Do not forget this parameter when changing transmission speeds.

The master can send a so-called broadcast frame to all the slaves. This frame is a write frame with the slave number equal to 0. The slaves do not reply to this frame. The write address is therefore the same in all the slaves (example : synchronisation frames). In this case a **Supplementary silence period after broadcast** (in milliseconds) may be necessary.

IV.5.3- Communication alarms

Parameters Page : a Jbus master channel can generate a communication alarm :

- **if there is no reply from any slave** : the defect must **be present for at least N ms** (N being between 0 and 60000 ms) before generating the alarm.
- **as soon as a single slave stops replying** : The LT cancels the alarm after a settable **maintain period**. This alarm is re-triggered even if any other slave is noted as absent before the end of the maintain period.

These 2 alarms are exclusive. The selected alarm type is included in the general synthesis of the central unit communication alarm.

Data Page : adds a 1 data bit: *Alarm bit for terminal block x channel y (protocol)*.

IV.5.4- Diagnostics / Counters

Parameters Page : Check the checkbox to place the counters in the internal data list.

Data Page : adds a 9 word data labelled *Frame counters Terminal block x channel y (protocol)* :

Rang	Signification
0	Number of complete correct exchanges (question + reply)
1	Number of replies received with CRC16 errors
2	Number of exception replies returned by the slaves
3	Number of LT transmitted commands
4	Number of LT broadcast commands transmitted
5	unused
6	Number of "slave not ready" replies received by the LT
7	Number of erroneous characters received by the LT
8	Number of commands re-triggered by the LT

IV.6- Setting the FIP channel parameters (on the LT100 and LT80)

Only the **station number** is configured in the Parameters page. The number should be between 1 and 255.

Warning : only the LT may have the **station** role on a FIP network. It cannot be the bus arbitrator. The FIP variables, which are to be produced or used by the LT, must be defined in the Data page corresponding to the channel.

The principles of the FIP protocol are described in the "Data" chapter .

IV.7- Setting the Ethernet channel parameters (on the LT100 and LT80)

IP parameters: The LT ignores the IP parameters by default and uses a BOOTP server, which attributes an available IP address to the LT. When the "The LT will use the BOOTP for IP parameters" function is unchecked the LT will take into account the IP parameters of the OPAL configuration, the details of which are as follows :

- **IP address:** identifies the network and node on a TCP/IP network. The address is generally given on 4 bytes separated by a dot (for instance, 87.34.53.12). Each number must be between 0 and 255. 1, 2 or 3 bytes designate the network and 3, 2 or 1 bytes respectively designate the node according to the address class A,B or C. The IP address by default is 255.255.255.255. The LT will, in this case, ignore the other parameters and use a BOOTP address server : it will return an available IP address to the LT.
- **Sub-net mask:** : address mask, which enables the display of the IP address division as a sub-net address and local address. This 32 bit mask only contains 1's for all the sub-net address parts of the complete IP address. The local address parts consist of 0's..
- **Gateway address:** the gateway IP address is to be found in the sub-net. If the LT is to communicate outside of the network to which it belongs then it must address this gateway. The default address is 127.0.0.1 and identifies the LT itself (not the gateway).
- **LT name :** name of the LT.
- **DNS address name :** name of the server IP address DNS (*Domain Name Server*). This server returns an IP address for a name, which designates a device or server on the TCP/IP network.

Parameter formats:

- **IPAddress :** xxx.xxx.xxx.xxx with xxx [0..255]
- **Sub-net mask:** xxx.xxx.xxx.xxx with xxx [0..255]
- **Gateway address:** xxx.xxx.xxx.xxx with xxx [0..255]
- **LT name:** maximum of 10 alphanumeric characters.
- **DNS name address:** xxx.xxx.xxx.xxx with xxx [0..255]

Once the parameters have been correctly entered the Ethernet link will support **TCP/IP** :

- **TCP/IP** : A set of industry protocol standards, which enable communication within a heterogeneous environment. The transport layer protocol supplies a protocol for managing enterprise routable networks as well as Internet access. It also contains the Session layer protocols. The basic TCP/IP protocols are, including the TCP protocol (Transmission Control Protocol), the IP protocol (Internet Protocol), the UDP protocol (User Datagram Protocol), the ARP protocol (Address Resolution Protocol) and the ICMP protocol (Internet Control Message Protocol). This protocol suite over that of the Ethernet determines the computer communication mode and inter-network connection procedures.

IV.7- Setting binary input parameters (TOR)

The following functions may be applied to a digital input :

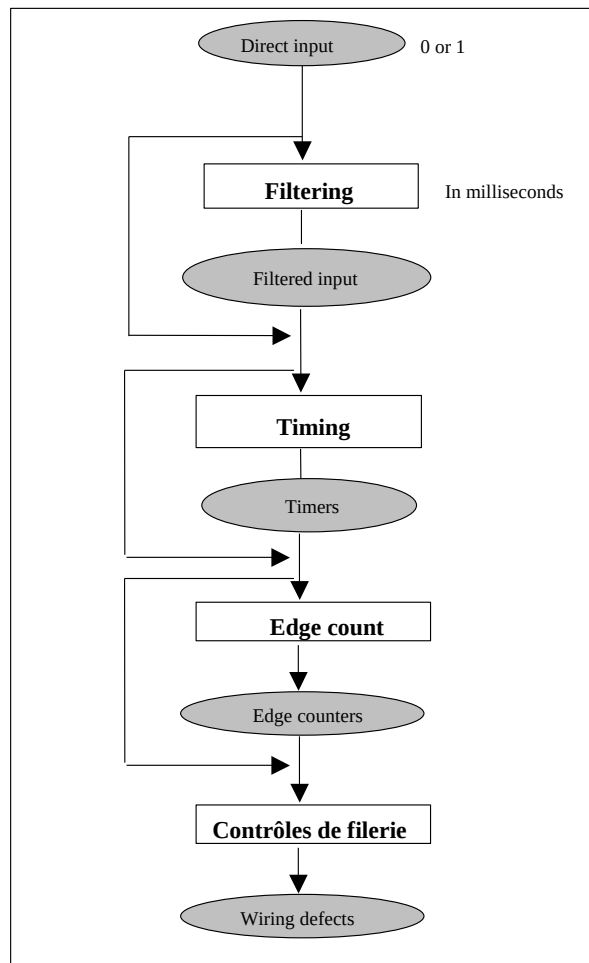


Figure 4 : Processes applied to a digital input

Setting the digital inputs board, without any specific parameter setting, leads to the creation in the Data Page of a Data with N bits corresponding to the N inputs of the board. This data corresponds to the direct inputs on which the following parameter settings may be undertaken :

- filters,
- timers,
- front counters,
- wiring controls (DI312 solely).

Each of these parameter settings lead to the creation of the specific data in the Data Page..

IV.7.1- Direct inputs

Data Page : The selection of N binary inputs in the Equipment Page automatically reserves an N bit data labelled *Block x : Digital Inputs*.

IV.7.2- Filters

A filtered input is an input whose status has been confirmed for duration of D. The duration D is the same for the rising edge and falling edge filtering.

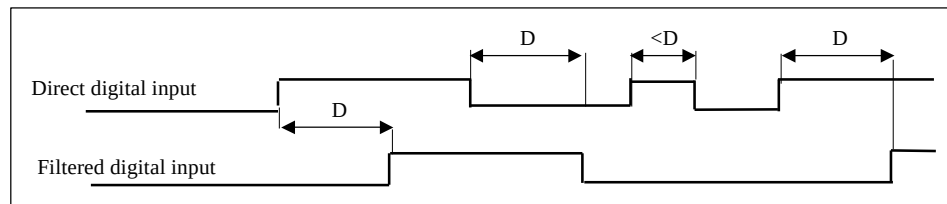


Figure 5 : Principle of Digital input filtering

Parameters Page : enter the value D for each input. A 5 digit entry is authorised. Scrolling the list gives access to all the inputs. The "Duration" button enables the duplication of the current value to all the other inputs.

Data Page : if at least one duration is positive OPAL reserves 1 Filtered digital input bit : 1 bit per channel. If all the filtering durations are identical then OPAL will reserve a single word D. Otherwise OPAL reserves one word per channel.

IV.7.3- Timers

The LT can affect exclusively to each channel :

- 1 to the last status timing (instantaneous timing),
- 0 to the last status timing (instantaneous timing),
- all status durations to 1 as from the last reset to 0 by a network master (cumulated to 1)
- all status durations to 0 as from the last reset to 0 by a network master (cumulated to 0)

The timers are updated **at each cycle**. The timers **are not saved** on power cuts. The unit for presentation/display can be chosen from (1/10s, s, mn, h).

Parameters Page : choose :

- timer type(-, instantaneous, cumulated)
- the level to time (0, 1)
- the timer unit (1/10s, s, mn, h)

The column header buttons enable the duplication of the current cell to the complete column.

Data Page : OPAL reserves 1 word per digital input to be timed, if at least one timer is declared.

IV.7.4- Edge Counters

The LT can count the change in status of the inputs to rising edge and to falling edge. The count is authorised by an authorisation bit per channel. The counter passes from 65535 to 0 in a cyclic manner. For those channels with set parameters the count starts at initialisation. The other channels may be validated later by writing the authorisation bit over the network.

Parameters Page : authorise the count (YES) on the Rising or Descending edge for each.

Data Page : N is the number of inputs which may be dispersed over numerous blocks. If at least 1 rising (respectively descending) edge counter is set then OPAL reserves:

- 1 rising edge count word per digital input,
- respectively : 1 falling edge count word per digital input.

IV.7.5- Wiring control (DI312 modules solely)

The **DI312 modules** have a settable comparison mechanism for **controlling the wiring** of the captors by connection of a 2 resistance network : **security inputs**. The resistance networks are reduced to 2 types : assembly in series (the 2 resistances in series) and the **parallel assembly** (the 2 resistances in parallel). The series resistance is always present. In the parallel assembly the captor is in series with the R_p which it eliminates on opening. In the series assembly the captor is parallel to the R_p , which it eliminates on close.

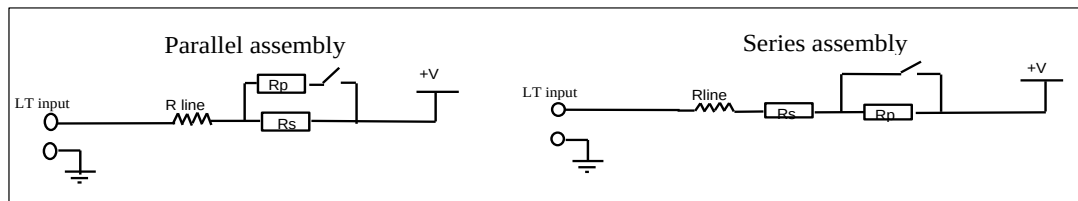


Figure 6 : Wiring security inputs

OPAL proposes the indication of the resistance equivalent to that of the network resistances (**Rcno**) when the captor is **normally open** and (**Rcnf**) when the captor is **normally closed**. The resistance values are in OHMS.

Parallel assembly	Series assembly
$Rcnf = R_s / R_p + R_{ligne}$	$Rcnf = R_s + R_{ligne}$
$Rcno = R_s + R_{ligne}$	$Rcno = R_s + R_p + R_{ligne}$

WARNING : The **settings are unique** for a **DI312 board** and therefore the same for all channels of the same DI312 module.

Parameters Page :

- **Control ?** : input may be used as a "simple" input by invalidating the wiring control. The validation/invalidation of a wiring control channel is possible over the network by writing a bit in the *Digital security inputs Masks* data.
- **RcNO, RcNF, Rligne** : The resistance values are to be set in OHMS.

Data Page : per 32 channel security board :

- 32 bits security mask : network writable bits
 - bit to 0 : the input is a security input
 - bit to 1 : the security input does not indicate an error
- 32 bits inputs status
- 32 bits of security inputs alarms

The status bit and the alarm bit code 4 possible states for each channel:

Input Status (LED green)	Alarm (LED red)	Description
0 (extinguished)	0 (extinguished)	Captor open normally
1 (lit)	0 (extinguished)	Captor closed normally
0 (extinguished)	1 (lit)	Input not connected or short circuit at 0V
1 (lit)	1 (lit)	Short circuit at +V

Resistances constraints :

- The resistances must have a maximum tolerance of 1%.
- $0.7 \text{ k}\Omega < \mathbf{Rcno} < 22 \text{ k}\Omega$
- $Rcno$ feeds the current to the measuring mechanism: $I \text{ (mA)} = 22 \text{ (V)} / (1 + Rcno \text{ (k}\Omega))$ where I must be between 1 mA and 9.96 mA . If I is calculated at greater than 9.96 mA, saturate it at 9.96 mA.
- $(2.95 \text{ (V)} / I \text{ (mA)}) < \mathbf{Rcnf} \text{ (k}\Omega) < Rcno \text{ (k}\Omega) - R_{ligne} \text{ (k}\Omega) - 1.95 \text{ (V)} / I \text{ (mA)}$
 - $\mathbf{Rligne} < 0.2 \text{ k}\Omega$

IV.8- Digital output parameter settings

The following functions can be applied to a digital output :

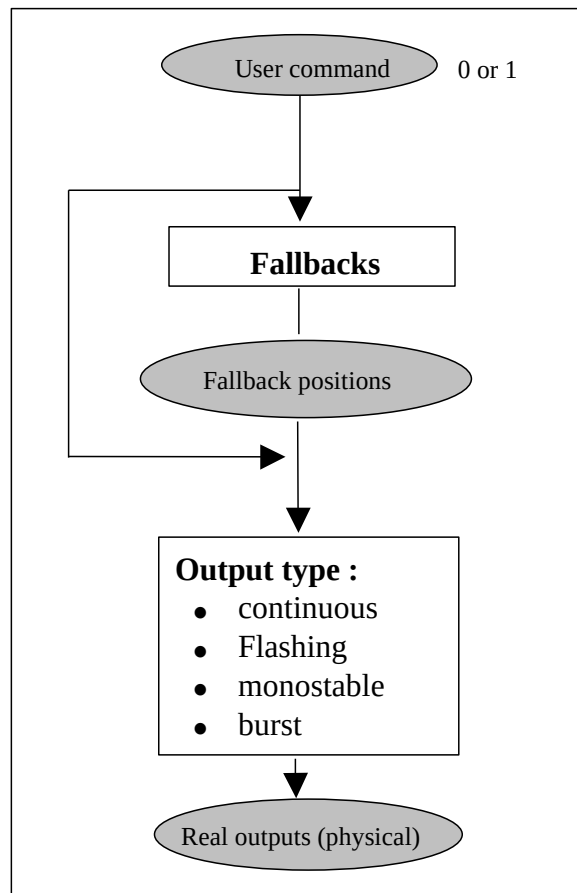


Figure 7 : Processes applied to digital outputs

Setting the digital output board, without any specific parameter setting, leads to the creation in the Data Page of a Data with N bits corresponding to the N outputs of the board. This data corresponds to the direct outputs on which the following parameter settings may be undertaken

- Signal type : continuous, monostable, flashing, burst,
- Fallback position.

Each of these parameters lead to the creation of the specific data in the Data Page.

IV.8.1- Command of digital outputs

Data Page : The selection of N digital output channels in the Equipment Page automatically reserves 2 data of N bits labelled :

- *Real digital outputs,*
- *Digital output commands.*

IV.8.2- Output signal processes

The LT can, from the same output command bit, **exclusively** produce the following output signals :

- **The continuous output equal to the command :**

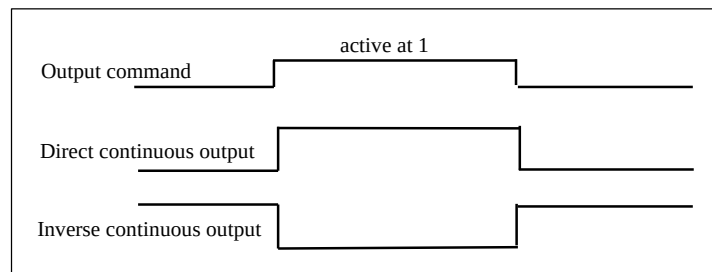


Figure 8 : Continuous digital output command

The Duration and $\frac{1}{2}$ Period settings are not important.

- **The flashing output :**

The output is flashing while the command is active. The cyclic ratio is 50-50. The DP duration of its half period is settable in milliseconds.

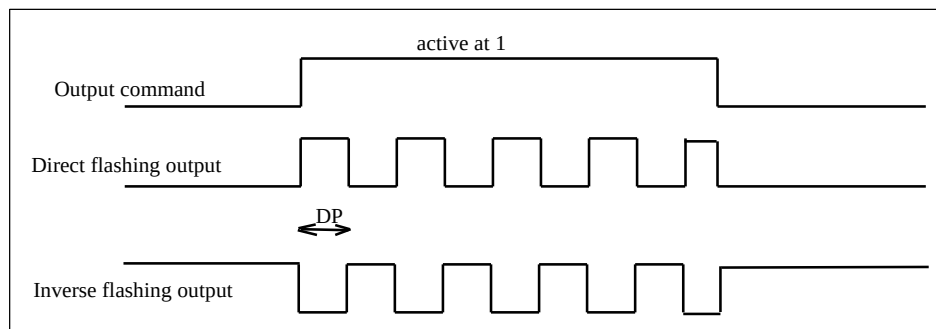


Figure 9 : Flashing digital output command

The duration parameter is not important.

- **The monostable output :**

The monostable output is active for a duration of D, settable in milliseconds, as from the rising edge of the command bit. Warning : the monostable duration is extended by D at each reactivation of the command bit.

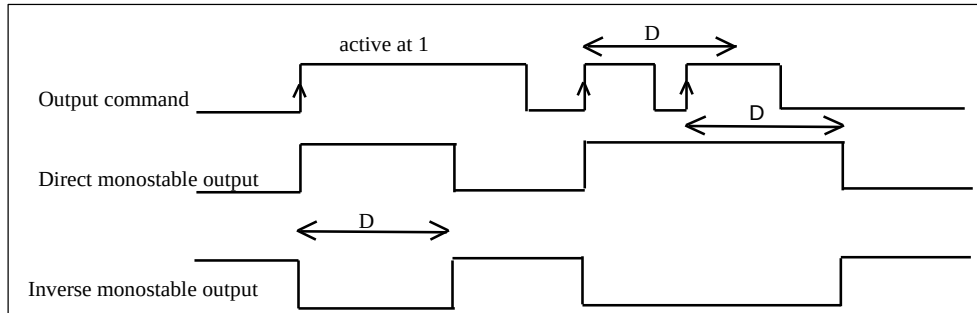


Figure 10 : Monostable digital output command

The `_Period` parameter has no signification.

- **The burst output:**

As from the rising edge of the command bit the burst output is flashing for a 1/2 period DP (in milliseconds) for monostable duration D (in milliseconds). Warning : the monostable duration is extended by D at each reactivation of the command bit.

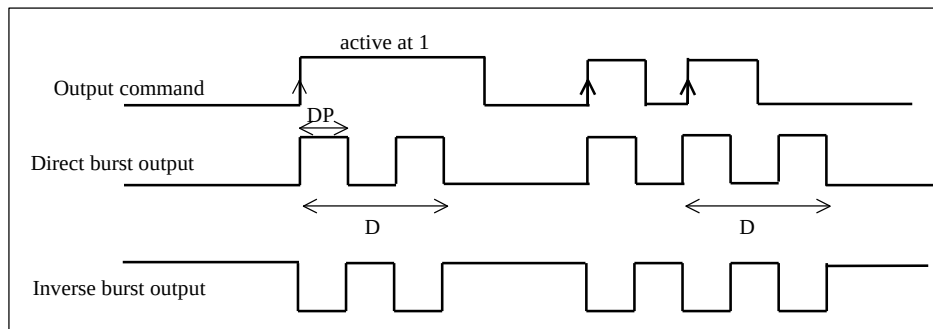


Figure 11 Burst digital output command

Summary :

Parameters Page: for each output channel, select

- The output type: Continuous, Monostable, Flashing or Burst
- The duration (D) and 1/2Period (DP) values if necessary
- The output polarity (direct, inverse)

The buttons at the top of the column enable the duplication of the current selection to the column.

Data Page: reserved data:

- N direct digital outputs bits
- N output commands bits

Also:

- If 1 of the outputs is monostable : N words corresponding to D
- If 1 of the outputs is flashing : N words corresponding to DP
- If 1 of the outputs is a burst : N words corresponding to D and N words corresponding to DP

IV.8.3- Processes / Fallbacks

The **general communication default** *Communication alarms synthesis* (refer to Central unit board parameter setting) controls the **passage of the outputs to a fallback position** (for those that were set with this function). The outputs switch into a fallback position on the **rising edge of the alarms appearance**. At the same time a memorisation bit for the fallback switch is set to 1. It should be reset to 0 either by the central unit, the network master or by using a process instruction as acknowledgement.

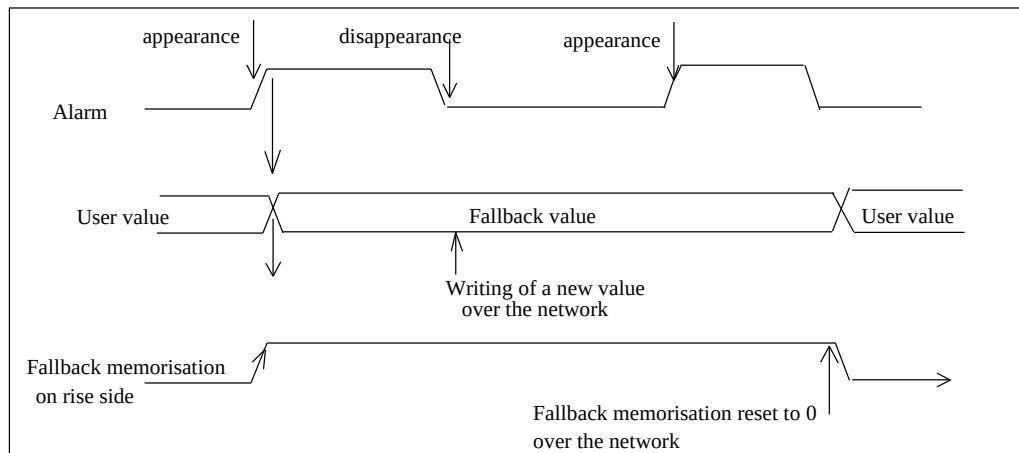


Figure 12 : Fallbacks on digital outputs

Parameters Page : Select the fallback position (0, 1, - (state)) for each output channel. The Fallback button enables the duplication of the current choice to all the channels.

Data Page : 1 general memorisation bit per fallback.

Per output : 1 direct digital output bit and 1 output command bit

IV.9- Setting parameters of analog inputs

The following functions may be applied to an analog input :

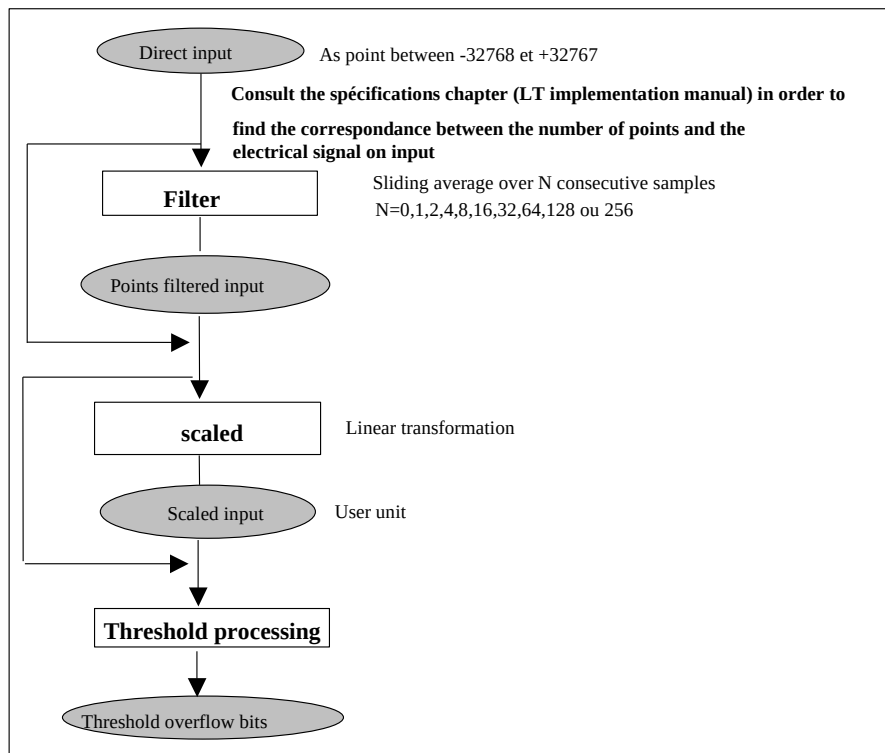


Figure 13 : Processes applicable to an analog input

Any parameter setting of the ANA inputs board, without any specific parameter settings, leads to the creation of a data of N words corresponding to the number N of board inputs in the Data Page. This data corresponds to the direct ANA inputs on which the following settings may be undertaken

- filtering,
- scaling,
- threshold overflows.

Each of these settings lead to the creation of specific data in the Data Page.

IV.9.1- Direct analog inputs

Each analog input channel reserves **one 16 bit word per channel** : "Block x : ANA Inputs ". For the LT, whatever the analog board, the direct analog inputs are digital values between -32768 and +32767 points.

Conversion Table current/voltage /temperature => number of points	
Non isolated current inputs (*)	$\pm 21,1\text{mA} \Rightarrow \pm 32767$ points.
Non isolated voltage inputs (**)	$\pm 10,25\text{V} \Rightarrow \pm 32767$ points.
Isolated voltage inputs	0 à 10,25V $\Rightarrow 32767$ points.
Isolated current inputs	0 à 21,1mA $\Rightarrow 32767$ points.
(*) Non isolated current inputs on AIO320	$\pm 20\text{mA} \Rightarrow \pm 32767$ points.
(**) Non isolated voltage inputs on AIO320	$\pm 10\text{V} \Rightarrow \pm 32767$ points.
PT100 sensor inputs on AIO320	-50°C $\Rightarrow$ -500 points ; +350°C $\Rightarrow$ 3500 points

IV.9.2- Filters

A filtered analog input is the result of a **sliding average on N consecutive samples**. N belongs to the following list {0, 1, 2, 4, 8, 16, 32, 64, 128, 256}. A number of samples equal to 0 deactivates the filtering. The samples are set to 0 at initialisation. The LT acquires **1 sample per cycle** and supplies **1 filtered analog value per cycle**.

Example : if an analog input presents a scale(rising edge) with a height H, the filtered input will progress by steps to the height H/N and with a duration equal to the cycle time Cy of the LT. The filtered input will be equal to the real input after a period equalling $N \times Cy$.

Parameter pages : Choose N in the list of samples for each analog input. The "Sample" button at the top of the column duplicates the current selection to the complete column..

Data Pages: N is the number of analog inputs. If at least 1 filter is different to 0 then OPAL reserves N words: *Block x : Filtered ANA inputs*.

IV.9.3- Scaling

Scaling consists of a **linear transformation of the acquired value**, eventually filtered, to another operator selected value. This transformation is a straight line defined by **2 points (A, A')** and **(B, B')** according to the diagram below. If the acquired value is less than A the transformed value will be saturated at A'. Inversely, if the acquired value is greater than B the transformed value will be saturated at B'. This transformation only manipulates whole numbers : adjust the units if necessary.

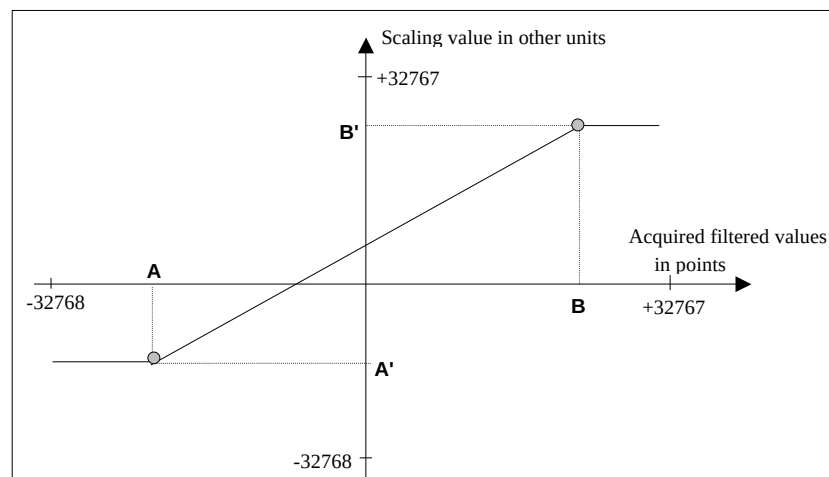


Figure 14 : Scaling of an analog input

Scaling example : A captor sends **0V for 5000 mbars** and **10V for 10000 mbars**. The LT AI210 module converts **0V to 0 points** and 10.25v to +32767 points, that is **31968 points for 10 V**. The scaling will therefore be : **A=0, A'=5000** and **B=31968, B'=10000**.

Remark about the parameter setting (A, A') and (B, B') : if the 2 points have the co-ordinates (-32768,-32768) and (+32767,+32767), the transformation is not undertaken. This corresponds to the straight line by default : $x=f(x)$ and the default parameter setting proposed by OPAL. However, the parameter setting of a straight line using the points (-1000,-1000) (+1000,+1000) is, mathematically, the same straight unit apart from the clipping but is not considered as such by OPAL and the scaled channel data will be created.

Parameter page : Enter the co-ordinates of the 2 points.
Any entered value is adapted to the [-32768, +32767] range.

Data Pages : N is the number of analog inputs If at least 1 input has a scale the OPAL reserves N words : *Block x : Scaled ANA inputs*.

IV.9.4- Thresholds

OPAL allows the setting of 2 exclusive threshold type parameters per channel :

⇒ **upper/lower threshold type**: this choice reserves 2 overflow bits per channel :

- The overflow bit of the lower threshold is set to 1 when the measure (scaled analog input) is less than the lower threshold.
- The overflow bit of the upper threshold is a bit set to 1 when the measure (filtered scaled analog input) is greater than the upper threshold..

Remark : the threshold values reference the scaled input values. Enter the values consequently.

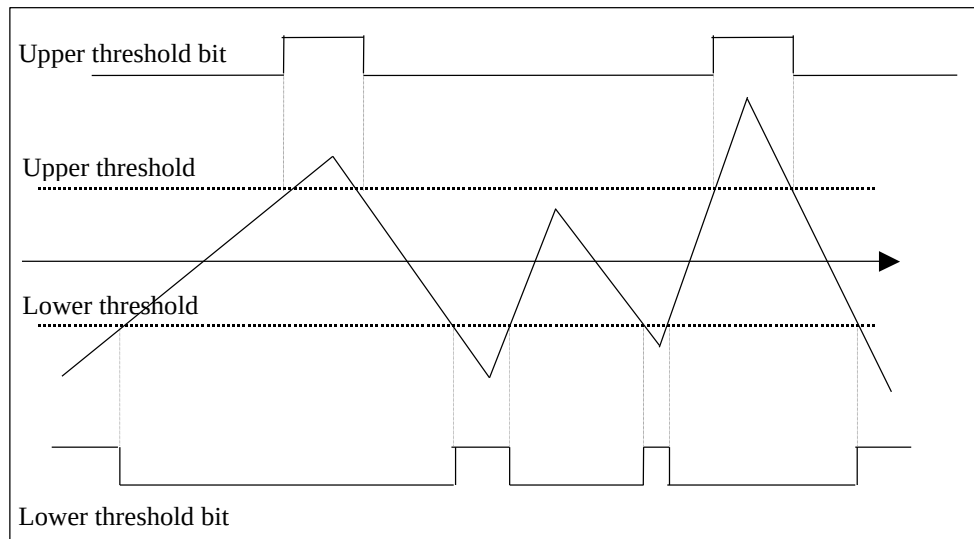


Figure 15 : Overflow of upper/lower type thresholds

⇒ **the Hysteresis type** : this choice reserves 1 bit per channel : The bit switches to 1 if the measure (scaled analog input) is greater than the upper threshold and this same bit will be reset to 0 if the measure (scaled filtered analog input) becomes less than the lower threshold.

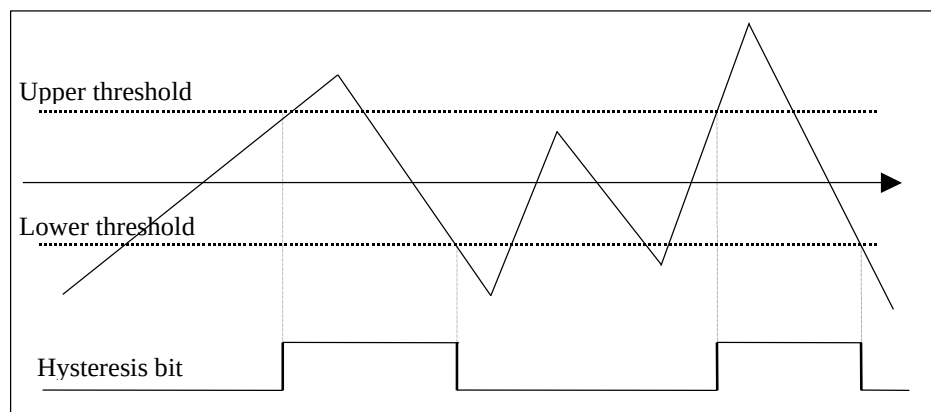


Figure 16 : Overflows of hysteresis type thresholds

Parameter Page : select the threshold type. Enter the threshold values according to the scale.

Data Pages: N being the number of analog inputs :

If a Upper/Lower type threshold has been selected then OPAL reserves N overflow bits per upper and lower threshold. If a Hysteresis type has been selected OPAL reserves N hysteresis bits.

IV.10- Setting analog output parameters

The following functions can be applied to an analog output :

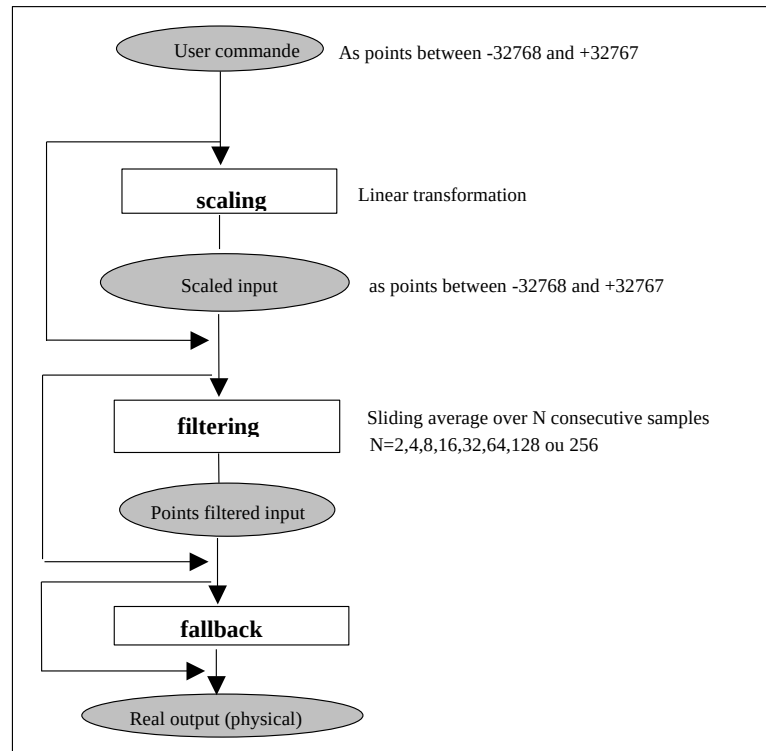


Figure 17 : Processes applicable to an analog output

IV.10.1- Analog output command

Data Page : Each analog output channel reserves a **16 bit word per channel** : "*Block x : ANA outputs*". The command value must be written in the data and within the range of -32768 to +32767 points.

Conversion table number of points => current/voltage	
Current outputs	0 / 32767 points => 4 / 20mA
Voltage outputs	± 32767 points => $\pm 10V$

NB : it is not necessary to have the command status and real state of the analog outputs.

IV.10.2- Scaling

Scaling consists of a **linear transformation of the user command value**, to another effective points value. This transformation is a straight line defined **2 points (A, A')** and **(B, B')** in the same manner as that for the inputs. If the command value is less than A the transformed values will be saturated at A'. Inversely, if the command value is greater than B the transformed value will be saturated at B'. This transformation only manipulates whole numbers : adjust the units if necessary.

Remark about the parameter setting (A, A') and (B, B') : if the 2 points have the co-ordinates (-32768,-32768) and (+32767,+32767), the transformation is not undertaken. This corresponds to the straight line by default : $x=f(x)$ and the default parameter setting proposed by OPAL. However, the parameter setting of a straight line using the points (-1000,-1000) (+1000,+1000) is, mathematically, the same straight unit apart from the values clipping but is not considered as such by OPAL and the scaled channel data will be created.

Parameter page : Enter the co-ordinates of the 2 points.
Any entered value is reduced to the [-32768, +32767] range.

Data Pages : N is the number of analog outputs. If at least 1 output has a scale then OPAL reserves N words
: *Block x : Scaled ANA outputs.*

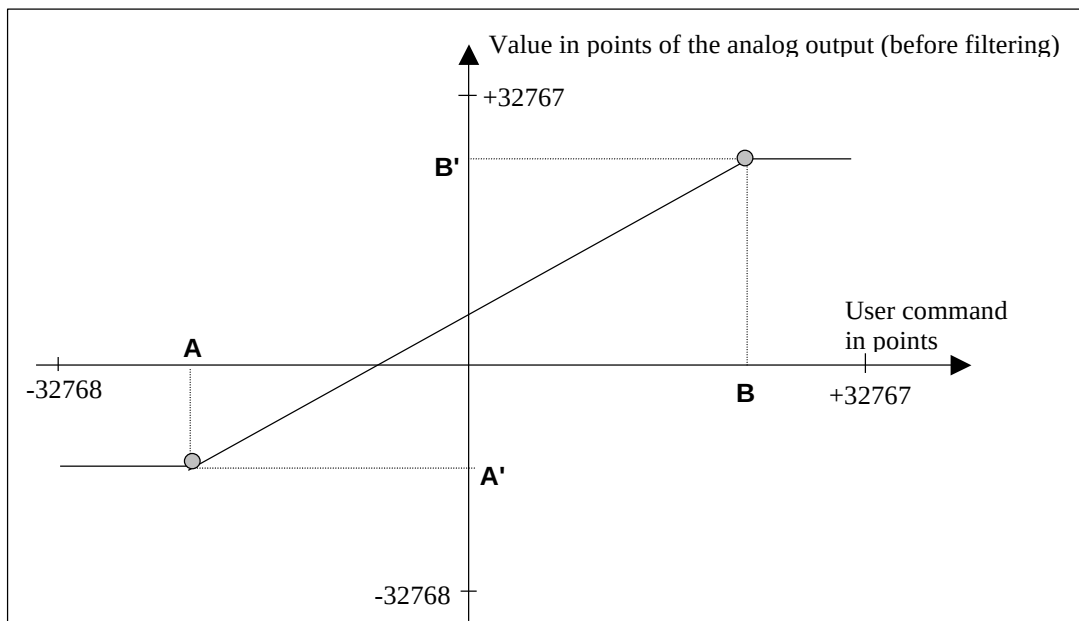


Figure 18 : Scaling of an analog output

IV.10.3- Filters

The filtering principle is the same as the analog inputs : the actual applied output is a sliding average of N successive commands. This average enables avoiding the "one offs" on the commands. The filtering is applied to the command value of the possibly filtered ANA output.

Parameter Pages : Select N in the samples list for each analog output. The "Sample button at the top of the column enables the duplication of the current selection to the whole column.

Data Pages : N being the number of analog outputs. If there is at least one filter different to 0 then OPAL reserves N words: *Block x : Filtered ANA outputs.*

IV.10.4- Fallback

The fallback principle is identical to that of the digital outputs

Parameter page : Choose whether the output has a fallback position (Yes/No) for each output channel and the fallback position [-32768; +32767]. The "Fallback" button enables duplication of the current value to the whole column.

Data Page : 1 general fallback memorisation bit: *"Fallback memorisation on ANA outputs".*

V- DATA PAGE

The Data page enables the visualisation of the **LT public Data List**. This is all the data created when configuring the equipment on the Equipment page and the selections of the Parameters page.

The down arrow (upper right) also proposes other data lists :

- the **list of all the LT's data**, always present, grouping together all the available data on an LT,
- if you have set the parameters of a CPU integrating an **FIP channel**, the **data access table via terminal block 1 channel 0 (FIP)**, present by default, groups together all the data accessible via this channel.
- If you have set the parameters of a CPU which integrates an **Ethernet channel**, the **data access table via the terminal block 1 channel 0 (OpenModbus/TCP) and the data access table via the terminal block 1 channel 0 (SNMP)**, present by default, group together all the data accessible via this channel.
- The **data access table via Terminal block 1 Channel 1 (Jbus slave)**, present by default, groups together the data accessible on Com 1 of terminal block 1 via a Jbus/Modbus master,
- If the parameter setting has been carried out the **data access table (or tables) via terminal block x channel y (protocol)**, present by default, regroups together the accessible data on terminal block x channel y via the specified protocol,
- If your central unit has **battery-backed memory (RAM)**, the **access table to protected data**, groups together the list of public data and enables the specification of the data to be saved and the backup mode.

V.1- The public data lists

This list is constituted **as the parameter setting progresses**. It may contain unused spaces if the selected modules or parameters have since been deleted. The data presented in this list is accessible via the network and the network addresses are found in the access tables of the chosen COMx.y. The list of public data is a sub-set of the complete LT data.

The list of public data consists of numerous columns :

- **I S** : **I** signifies LT **internal data** (without signification in the public data list), **S** signifies **backed up data**.
- **Name** : Name defined by default. This name can be modified by the operator using the Modify function (Modify button) ,
- **Data type**: BIT or WORD (reminder : 1 word = 16 bits),
- **Length** : number of data of the associated type,
- **Description to be given in the data list** : a data comment.

This list may be :

- Completed with supplementary user created data,
- Modified to propose a different data layout available to the communication channels.

V.2- List of all the LT data

The LT disposes of specific data, which can be used for the OPAL processes, for each declared input/output board. This data is visible in the "**list of all LT data**" and identified by **BLOCKx**, x being the rack order of the board [1..15]. The supply data block is called **SUPPLY** and the central unit is called **CPU**.

The Index presented for each data in the list is **reserved** for the internal LT operation and is not useable by an operator.

Here is the list of available data and its type for each type of board :

Examples :

- for a **PSDxxx** the "Status" data is in the word SUPPLY[0]
- for a **CPUxxx** the "Cycle time" data is in the word CPU[9]
- for the **DI310 in position 5** the " 32 digital input values array " is in the word BLOCK5[9]

The data structure associated with a **PSDxxx** board consists of 3 words :

- Board status
- Initialisations counter
- Supply command

The data structure associated with a **CPUxxx** board consists of 13 words :

- Millisecond counter
- Cycle counter
- Declared input/output boards : the configuration analysis report. 1 bit per input/output : The first input/output board is the CPU. If the input/output board was declared in the configuration then the corresponding bit is at 1.
- Real input output/boards: report of the real presence of the input/output cards. The card status is verified at each LT cycle. If the type and board order is correct then the bit is set to 1..
- Initialised Input/Output boards : some boards require numerous LT cycles to become initialised. The board is operational when the bit corresponds to 1.
- Input/output LEDs refreshing: the corresponding bit is set to 1 if the LEDs refresh was undertaken correctly.
- Board status
- LT operating modes command
- LT operating modes time command
- Cycle time
- Max. cycle time.
- Min. cycle time.
- CU LEDs commands

The data structure associated with a **DI310** board consists of 7 words :

- Board status (1 word)
- Board initialisations counter (1 word)
- 32 digital input values array (2 words)
- Board 32 led command (2 words)
- Command for the threshold detection supply (1 word)

The data structure associated with a **DO310** board consists of 6 words :

- Board status (1 word)
- Board initialisations counter (1 word)
- 32 digital input values array (2 words)
- Board 32 led command (2 words)

The data structure associated with a **DI410** board consists of 10 words :

- Board status (1 word)
- Board initialisations counter (1 word)
- 64 digital input values array (4 words)
- Board 64 led command (4 words)

The structure associated with a **DI312** board consists of 13 words :

- Board status (1 word)
- Board initialisations counter (1 word)
- The 32 digital input values array (2 words)
- The 32 digital input status array (2 words)
- The 32 digital input status mask array (2 words)
- Command of the boards 32 green LEDs and 32 red LEDs (2 words)
- Command table of the defect thresholds and status (3 words)

The data structure associated with a **DIO210** board consists of 5 words de 5 mots :

- Board status (1 word)
- Board initialisations counter (1 word)
- 16 input values array (1 word)
- Table of the 8 digital output values and command of the 8 red output LEDs of the board (1 word)
- Command of the boards 16 green output LEDs (1 word)

The data structure associated with an **AI110** board consists of 11 words :

- Board status (1 word)
- Board initialisations counter (1 word)
- 8 ANA input values array (8 words)
- Command of the 8 red LEDs and 8 green LEDs (1 word)

The data structure associated with an **AI210** board consists of 20 words :

- Board status (1 word)
- Board initialisations counter (1 word)
- 16 ANA input values array (16 words)
- Command of the 16 red LEDs and 16 green LEDs (2 words)

The data structure associated with an **AO120** board consists of 11 words :

- Board status (1 word)
- Board initialisations counter (1 word)
- 8 ANA output values array (8 words)
- Command of the 8 red LEDs and 8 green LEDs (1 word)

The data structure associated with an **AO220** board consists of 20 words :

- Board status (1 word)
- Board initialisations counter (1 word)
- 16 ANA output values array (16 words)
- Command of the 16 red LEDs and 16 green LEDs (2 words)

The data structure associated with an **AIO320** board consists of 18 words :

- Board status (1 word)
- Board initialisations counter (1 word)
- 8 ANA input values array (8 words)
- 4 ANA output values array (4 words)
- Block error counter: count of erroneous messages (1 word)
- Command of 8 orange LEDs (1 word)
- The 8 bit array of the ANA inputs upper overflow (1 word)
- The 8 bit array of the ANA inputs lower overflow (1 word)

V.3- Operations on the data lists

These operations only concern the public data list and the LT data list. All of the following described operations are accessible in the Data menu. The Add, Modify and Move manipulations are directly accessible with the buttons present.

Beware of the consequences of these operations on the processes.

Add/create : The creation window requests the operator for the name of the data to be added, the type, the length in bits or words and the list commentary label. The addition is placed at the bottom of the public data list.

Move : click on the data to be moved with the mouse pointer : the data colour is inverted (white on blue). Move the mouse to the desired new data position. Once the mouse pointer has the form of a hand , click to move the data. A confirmation window informs you of the move prior to moving.

Modify : Enables modification of the name, the type, the data length or commentary. Only the name can be modified for OPAL created data.

Delete : A delete can only be undertaken on user created data.

Optimise : Optimisation eliminates unused space and **reorganises all the data lists** by placing the bits at the top of the list followed by the words.

V.4- Data access table via Terminal Block x Channel y (Modbus/Jbus slave)

The access table is deduced from the list of public data :

- Unused spaces at the end of the list are deleted,
- the **addresses** range from 0000 to FFFF in **hexadecimal**.

If the **data type is a WORD**, the address given in the 1st column is a **WORD address**.

If the **data type is a BIT**, The address given in the first column is a **BIT address**.

NB : The 0010 to 001F address bits are accessible by word at the 0001address. Inversely a 0002 address word may be read as 16 bits as from address 0020.

Example :

Address (hex)	Type	Length (decimal)	Label
0000	WORD	16	Block 1 : analog outputs
0010	WORD	16	Block 2 : analog inputs
0200	BIT	32	Block 3 : digital outputs
0022	WORD	16	Block 4 : analog inputs
0320	BIT	16	Central unit status

The 16 words of block 1 occupy the word addresses 0000 to 000F.

The 16 words of block 2 occupy the word addresses 0010 to 001F.

The 32 bits of block 3 occupy the 2 word addresses 0020 and 0021 or the 32 address bits 0200 to 021F.

The 16 words of block 4 occupy the word addresses 0022 to 0031.

The status of the central unit occupies the word address 0032 or the 16 bit addresses 0320 to 032F.

The **Accessible Limit** button fixes the address above which the data are no longer accessible via Jbus or Modbus requests. In order to set this limit, you have to click on the chosen variable and define it as the Accessible Limit by clicking on the Accessible Limit button. The variables whose addresses are greater than the upper limit set by the chosen variable will be inaccessible.

Accessible data limits through ModBus/JBus slave channel

- a word address is limited to the value 7FFF
- a bit address is limited to the value 7FFF

For anything above, the addresses are inaccessible and the data cannot be used in the processes.

NB : if you have an important number of data it is recommended that you optimise the addressing in the data list by placing the bit data types before the word data types.

The **Base address** button enables the redefinition of the addresses of the word and bit databases. The data from each is, by default, accessible from the address 0.

V.5- Data access table via terminal Block x Channel y (Modbus/Jbus master)

A **general comment** about digital value entry in hexadecimal format : the entry in a field may be given in decimal or hexadecimal. In order to change the base, point at the data to be entered and click on the **right-hand mouse button**. A hexadecimal/decimal menu will appear. Click on the chosen option.

Reminder : this data is accessible from the data access tables via **COMx.y (Jbus or Modbus master)**.

This access table is empty by default. Using the Add button does the parameter setting of a Modbus/Jbus master frame (request). The **Modify** and **Delete** operations are also available.

V.5.1- Adding or modifying a frame

Press the Add key to declare a new frame and fill in the window. The frame is added to the list of data assigned to this Jbus master channel after declaration.

The frame parameter setting is undertaken by entering the following parameters :

- **slave number** [0..255] (0 is only used for broadcast frames),
- **function** (refer to following paragraph),
- **Jbus address**: address circulating on the network and to which one desires writing or reading from the slave,
- **Jbus length** : length, in bits or words, of the data to be read or written,
- **Exchange variable**: data from the public data list that will be transmitted (write) or received (read). This data may be selected directly in the list of available data with the Change button. It fixes the initial address of the data concerned in the LT,
- **Conditional triggering**: enables triggering the present frame if the associated **bit is equal to 1** . The available associated data in the list and accessible via the Change button are exclusively of type bit ; **the bit is reset to zero once the frame launched**,
- **report** : the result or **status** of the frame may be placed exclusively in a word type data.

NB : prepare the possible trigger bits and report words in the internal data list.

V.5.2- Implemented functions

The LT available Modbus/Jbus functions are

- Function 1 : Read N consecutive bits (N<2000)
- Function 2 : Read N consecutive input bits (N<2000)
- Function 3 : Read N consecutive words (N<125)
- Function 4 : Read N consecutive input words (N<125)
- Function 5 : Write 1 bit
- Function 6 : Write 1 word
- Function 7 : Fast read 8 bits (the address is predefined in the slave)
- Function 15 Write N consecutive bits (N = [1..1968])
- Function 16 : Read N consecutive words (N = [1..243])

Function 8 : diagnostics functions:

Sub function number	Name	Slave actions
0	Echo	The slave must return the echo of the transmitted word.
1	Re-initialisation	The slave must reconnect to the network and initialise its communication coupler
2	Echo	The slave must return the sent data
3		Not implemented
4	Disconnection	The slave must not reply. Cancelled via a sub-function 1
10	Diagnostics reset	The slave resets its diagnostics counters to 0
11 to 18 respectively	Reading of the diagnostics counter	The slave returns the content of counter 1 to 8 respectively

V.5.3- Conditional triggering and exchange reports

We have seen that the following can be associated with each frame :

- 1 frame trigger bit. If this bit is 1 at the analysis moment of the LT cycle request, the frame is transmitted. This trigger bit is reset to 0 by the LT after frame transmission. NB : if the trigger bit is not set it is then forced to 1 permanently.
- 1 report word of the slave response :

List of Modbus/Jbus error codes (communication status):

Decimal	Hex	Comment
0	0	Exchange underway
256	100	Exchange correct
769	301	Exception code : unknown function
770	302	Exception code: incorrect address
771	303	Exception code: invalid data
772	304	Exception code: slave not ready
773	305	Exception code: acknowledgement
774	306	Exception code: not acknowledged
775	307	Exception code: write error
776	308	Exception code : overlapping zones
896	380	Connection error
897	381	Connection warning
1024	400	Incorrect slave number
1025	401	Incorrect function code
1026	402	Incorrect length
1027	403	Incorrect sub-function code
1028	404	Incorrect address
1029	405	Incorrect data
1030	406	Incorrect frame length
1280	500	Slave absent
1281	501	CRC error
4096	1000	At transmission : frame underway
4097	1001	At transmission : broadcast error
4099	1004	At transmission: length error
4100	1005	At transmission: offset error
4101	1006	At transmission: function error
4102	1007	At transmission: sub function error
4103	1008	At transmission: sub function data error
4104	1009	At transmission: storage error

In bold the most frequently encountered communication status error codes.

V.5.4- Frame sequencing in the LT cycle

The LT cycle can be resumed globally as:

- Reading input boards
- Analysis and processing of the read frames
- Processes
- Writing to output cards
- Led refreshing
- Analysis and processing of the write frames
- LT inspection task
- ... return to start ...

The LT runs the following sequential algorithm at each cycle and for the Modbus/Jbus master frames :

```

If no current exchange running
Then
    If the pointed frame is authorised Then send End if
    Point the following frame
Else
    If exchange terminated Then
        Update the report (if set)
        Class the received data if read command
    Else
        If exchange is running Then
            Sending Frame or re-launch underway
            End if
        End if
    End if
End if
  
```

Note that the frame pointer advances 1 frame per cycle. Therefore if the frame is not authorised the LT doesn't send in that cycle.

V.6- Data Access table via Terminal Block 1 Channel 0 (FIP)

Only the **Station function** is implemented on the LT. The **bus arbitrator function** is not implemented. A station only manages the **FIP production/consumption variables**. A bus arbitrator also manages the access mechanisms to the communication mediums. One sole station disposes of the bus arbitration function on a FIP network at any given moment. The LT ensures the traffic of variables but not the messages traffic

The **silence time** = 240µs and the **return time** = 42 µs.

V.6.1- Parameter setting synthesis for the implementation of FIP variables

First of all, here are the elements which enable entering a FIP variable and decide whether it is produced or consumed. The FIP protocol mechanisms are detailed in the following chapters.

Equipment page : Declare a CPU board comprising of a FIP port (Terminal block 1 Com 1 obligatory).

Parameters page :

- Select the **FIP** protocol on **Com 1 of the terminal block 1**
- Just indicate the **station number** between 1 and 255.

Data Page :

- Go to the page "**Data access table via COM0.1 FIP**"
- Select the variable to be produced or consumed in the list (Beware : if the variable does not have a label it will not be visible!)
- Click the "**FIP Parameters**" button

The operator can declare

- **16 physical variables** in zone 1 identified as 0ixxh, with i from 0 to F and xx corresponding to the FIP station.
- the **free allocation variables** in zone 3 identified from 3000h to 7FFF. For these variables, one must take care not to use the same identifier for two different variables.

Setting the parameters consists therefore of declaring a relation between a FIP identifier and a variable on the list of internal LT data.

Fill in the window for each variable:

- variable identifier (use the right click to select the Hex or Decimal entry base)
- type of variable (refer to following paragraphs)
- number of bytes to be transported for the string types
- optional :
 - ⇒ for a produced variable :
 - the refreshment status period
 - ⇒ for a consumed variable :
 - the promptitude period
 - the name of the bit receiving the promptitude status
 - the name of the bit receiving the refreshment status

V.6.2- Variable identifiers

The FIP protocol manipulates variables identified by a code ranging from 0000h to FFFFh enabling the modelling of the FIP network as a base of 65536 distributed variables. Each variable has a producer station and one or more consumer stations. The maximum number of stations is 256 (00h to FFh; 0 not authorised by OPAL).

The identification of certain variables is linked to the station number : the **physical allocation**. Some of the physical variables are left for the user's usage (physical user variables Zone 1 of the FIP protocol), the others are used by the system to manage the network layers (SM_MPS variables: Zones 2 and 3 of the FIP protocol).

The identification of certain other variables is not linked to the station number : the **free allocation** (Zone 4 of the FIP protocol). The LT uses the identifiers in the following array (in bold) from the FIP network identifiers. Each LT automatically produces its own **"Presence" system variable**: identifier 14xx (xx = station number).

IdL= zone limit identifiers ; ID= specific identifiers

Zone	IdL	ID	Variable signification
1	0000 0FFF		User physical allocation : 16 variables per station The 16 variables assigned to the station XX are 0iXX with i from 0 to F
2	1000 10xx 14xx 1FFF		Physical allocation: 16 variables per station The 16 variables assigned to the station XX are 1iXX with i from 0 to F. Only those produced by the LT : Identification (byte string) Presence (type integer 16 bits)
3	2000 2FFF		Physical allocation: 16 variables per station The 16 variables assigned to the station XX are 2iXX with i from 0 to F. The LT doesn't use any
4	3000 7FFF		Free allocation of the application layer Identifiers positioned with the OPAL parameter setting The identifiers of this zone do not depend on a station! Check carefully that they are not used on other stations.
5	8000 8FFF		Physical allocation: 16 variables per station The 16 variables assigned to the station XX are 8iXX with i from 0 to F. Description variables of zone 1 variables 1 : the variable 8iXX describes the variable 0iXX
6	9000 9FFF		Global allocation (reserved)
7	A000 AFFF		Free allocation (unused by the LT)
8	B000 FFFF		Free allocation : description variables of zone 4 variables

V.6.3- Variable types

Under the FIP protocol an identifier corresponds for instance to the transport on the network of a **string of bytes** encapsulated in a frame. The same identifier can therefore enable the transport of a table of 8 whole measures in the form of a 16 byte string. The declaration of the **identifier type and number** must be **strictly identical at the producer and the consumer**.

The producer produces a FIP frame on the line. The arrangement of the bytes (possible inversion of the high/low order) depends on the type. The interpretation of the byte string is undertaken by the consumer at its convenience.

Example : A producer produces a double integer. The transport is effectuated in the form of a 4 byte string. A consumer can interpret it as a 32 bit array. It is important that the number of bytes is coherent.

Warning : The LT can only produce and consume a **maximum of 32 variables**. Therefore, in order to increase the useful network rate data must be grouped together as much as possible in the strings.

The LT authorises the following types of FIP variables :

Bytes	Type : possible interpretations	Commentary
1	Boolean	coded 00 or FF
	Short unsigned integer 8 bits	
	Sort signed integer 8 bits	
2	Unsigned integer 16 bits	
	Signed integer 16 bits	
4	Long unsigned integer 32 bits	
	Long signed integer 32 bits	
	IEEE floating simple precision	Not useable by the LT
8	IEEE floating double precision	Not useable by the LT
n	String nx8 bits	(1)
	String of n bytes	(1)
	String of n visible characters	(1)

(1) The maximum length of useful data is 126 bits consisting of :

- the actual variable
- the eventual refresh status associated with the variable and requested by the user (boolean=1 byte)

V.6.4- Variable production and consumption

A producer (respectively, a consumer) does not produce (respectively, does not consume) a variable directly on the network. This **rate is defined in the bus arbitrator**. It produces (i.e. Refreshes) a variable in its private buffer (or it consumes (ie . reads) a variable from its private buffer). The problem is therefore posed for the consumer as to the « freshness » of the exchanged data.

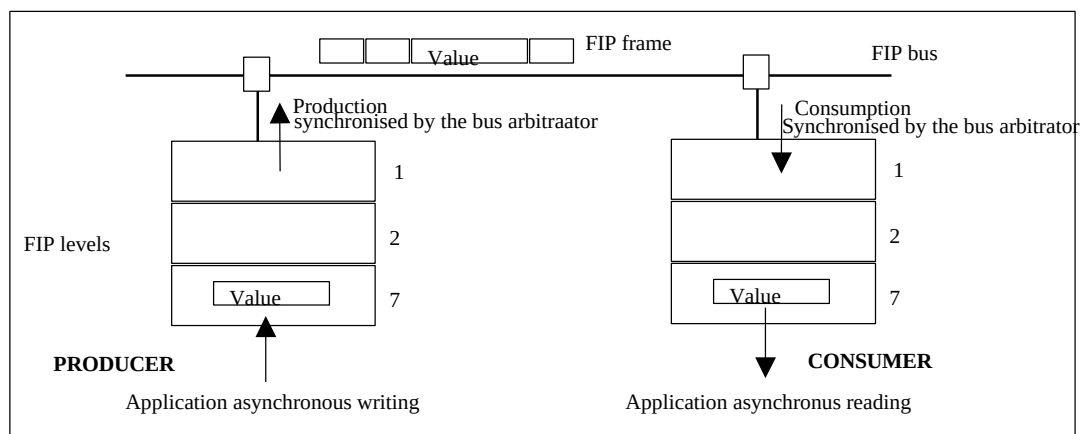


Figure 19 : Production and consumption mechanism of a FIP variable

The LT implements the **Asynchronous Refresh** and **Asynchronous Promptitude** mechanisms.

V.6.5- Variable refresh status

The refresh status of a variable (Boolean = 1 byte) is elaborated by the **FIP Application 7 layer of the producer**. Each time this variable is written to the private buffer the Application 7 layer sets a timer with the refresh period value (which can be set with OPAL). The Boolean remains true while the timer has not expired. Each consumer will therefore know if the variable producer has respected the production delay attached to the variable.

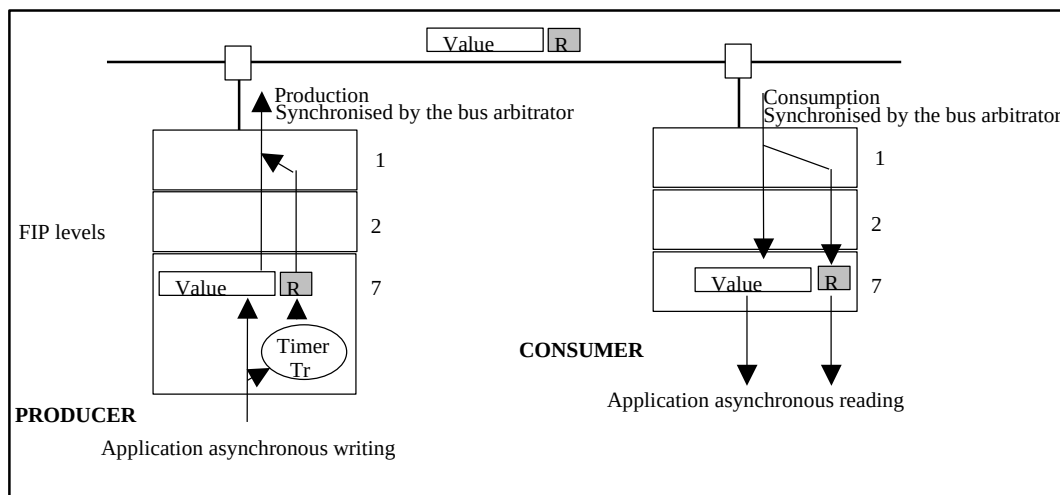


Figure 20 : Refresh status mechanism associated with a FIP variable

OPAL setting :

- of the producer station : refresh period T_r
- of the consumer station: Name of an internal LT data where the refresh status is stored.

V.6.6- Variable promptitude status

The variable Promptitude status (Boolean=1 byte) is elaborated by the the **FIP Application 7 layer of the consumer**. At each arrival of this variable in the private buffer the Application 7 layer sets the timer to the value of the promptitude period (which can be set with OPAL). The Boolean remains true while the timer has not expired. This Boolean is not transported by the network. Each consumer will know whether the bus arbitrator respects the call period of the variable on the network and if the producer and consumer buffer transfer functions correctly.

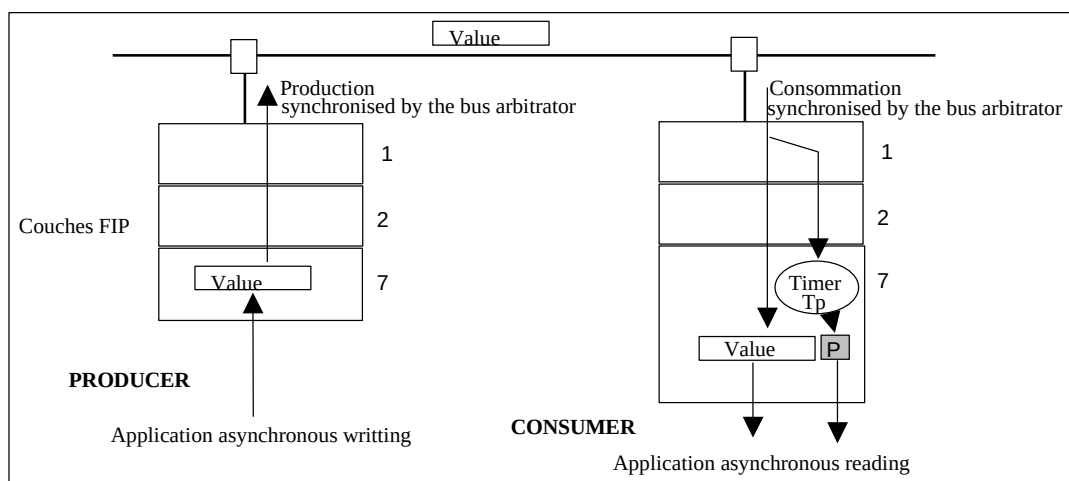


Figure 21 : Promptitude status mechanism associated with a FIP variable

OPAL settings:

- on the producer station : none
- on the consumer station: Promptitude period T_p and Name of a data internal to the LT where the promptitude status is stored .

V.7- Data Access table via Terminal block 1 Channel 0 (OpenModbus/TCP slave)

- **Open Modbus TCP Slave** : This protocol, defined by Modicon, consists of encapsulating the Modbus exchanges in the IP frames. It uses the TCP connected mode. It offers the same functionality as the « ModBus slave » channels on asynchronous links of the product. The differences with the Modbus protocol over asynchronous channel are as follows :
 - No slave number (between 1 and 255), as the addressing is undertaken with the IP address
 - Usage of the TCP connected mode. This allows simultaneous connections with numerous masters on the network. The limit is set to 2 masters, which corresponds to a process master and a diagnostics console master.

The access table is deduced from the public data list :

- The unused spaces at the end of the list are deleted,
- the **addresses range from 0000 to FFFF in hexadecimal.**

If the **data type is a WORD** then the given address in the 1st column is a **WORD address**.

If the **data type is a BIT** then the given address in the 1st column is a **BIT address**.

NB : The bit of address 0010 to 001F is accessible by word at address 0001. Inversely an address word 0002 may be read as 16 bits from address 0020.

Example :

Address (hex)	Type	Length (decimal)	Label
0000	WORD	16	Block 1 : analog outputs
0010	WORD	16	Block 2 : analog inputs
0200	BIT	32	Block 3 : digital inputs
0022	WORD	16	Block 4 : analog inputs
0320	BIT	16	Central unit status

The 16 words of block 1 occupy the word addresses 0000 to 000F.

The 16 words of block 2 occupy the word addresses 0010 to 001F.

The 32 bits of block 3 occupy the 2 word addresses 0020 and 0021 or the 32 bit addresses 0200 to 021F.

The 16 words of block 4 occupy the word addresses 0022 to 0031.

The central unit status occupies the word address 0032 or the 16 bit addresses 0320 to 032F.

The **Accessible Limit** button sets the address above which the data is no longer accessible via the network. The addresses are not accessible above this limit.

Data access limits via ModBus/JBus slave channel

- a word address is limited to a value of 7FFF
- a bit address is limited to a value of 7FFF

The addresses are inaccessible over these values and the data may not be used in the processes.

NB : If you have a large amount of important data, it is recommended that you optimise the addressing in the data list by placing the bit data types above the word data types.

The **Basic address** button enables the redefinition of the word and bit database addresses. By default the data for each is available at the address 0.

V.8- Data access table via Terminal block 1 Channel 0 (SNMP)

- **SNMP** : *Simple Network Management Protocol*. Internet standard for the administration of hosts, routers and other devices on the network..

All of the variables present in the **public data list** are available in the data access table via the terminal block 1 Channel 0 (SNMP).

The user disposes of 9 SNMP variable types to set the OPAL variables.

Variable TYPE	Length in bytes
Signed 32 bit integer	Fixed at 2
Byte string	free (2 by default)
NULL	free (2 by default)
OID	free (2 by default)
Sequence	free (2 by default)
IP address	Fixed at 2
Counter 32 bits	Fixed at 2
32 bit gauge	Fixed at 2
TimeTicks	Fixed at 2
Opaque	free (2 par default)
Counter 64	Fixed at 4

The Delete button allows the deletion of the SNMP parameters assigned to a variable.

VI- LT CONNECTED PAGE

The LT connected page proposes 5 services :

- Write to the LT the entered configuration,
- Refresh the equipment configuration,
- Dynamic display of the inputs/outputs status,
- Communication parameter setting,
- Launching Jbus master requests.

VI.1- Write configuration

This function enables the **downloading of the .OPA file** currently open on the PC **to the LT**. Check that the parameter setting of the PC serial link is identical to that of the LT communication port : this is the case if the LT was switched on in parameter setting mode (PRM).

The configuration loading is followed by the LT re-initialisation by OPAL (approx. 7 seconds).

OPAL does not allow the writing of the configuration to start if the LT is not connected, or inaccessible,. A window will display « No LT connected » as well as the size occupied, in bytes, by the entered configuration.

VI.2- Size of the entered configuration

The available memory space in the LT100 for the configuration backup is **4 Kbytes** (128 bytes are reserved by the embedded LT100 software).

The available memory space in the LT80 for the configuration backup is **2 Kbytes** (2 bytes are reserved by the embedded LT80 software).

The available memory space in the LT160 for the configuration backup is **4 Kbytes** (2 bytes are reserved by the embedded LT160 software).

OPAL supplies the following information before writing the configuration to the LT :

- **Available space** in bytes,
- **Used space** in bytes and in percentage,
- **Free space** in bytes and in percentage.

Therefore, according to the entered configuration, you know the influence of such and such a process on the size of your configuration.

VI.3- Refresh

Clicking on this button **refreshes the elements on the « LT connected » page** and updates the list of functional modules.

IMPORTANT : OPAL, however, does not recover the complete LT configuration and therefore does not update the other pages.

VI.4- Dynamic input/output blocks

The test panel of direct inputs or outputs appears by clicking on a block in the **inputs/outputs block list**, as well as the operational status of the module seen by the central unit : status of the boards and initialisations counter.

The test panels enable reading or directly forcing the outputs independently of the programmed processes in the Process Page.

Status of boards : The status of the operational module is given by a comment : correct, absent or by digital code, the signification of which is given below :

F	C	B	A	9	8	7	0
Position		Y	Wdg	X	Vcc		Board code

Position : value of 1 to F according to the position of the board on the base

Wdg Bit: value of 1 at rest, Wdg active at 0 generally linked to an internal power problem

Vcc Bit: value at 1 if the internal power of the board correct

Module	Bit Y	Bit X	Code
DI310	1	Al ext	03h
DI410	1	Al ext	06h
DI312	1	Al ext(*)	14h
DO310	Fault	Al ext	05h
DIO210	Monostable	VRel	16h
AI110	1	1	80h
AI210	1	1	81h
AO120	1	1	84h
AO220	1	1	85h
AIO320	Monostable	0	83h
NC100	1	1	30h to 37h

with :

- Al ext : value at 1 if the external power present on the terminal blocks is between $V_{pow} \pm 20\%$
- (*) Al ext of the DI312 : external power between $24V \pm 10\%$
- Fault : value at 0 if overload on a digital output
- the unused X and Y bits are by default at 1

Example : A faulty supply of the terminal block board associated with DI310 at position 5 gives (the status) 5D03h (bit X=0).

VI.5- Communication parameters

Selecting « **Communication Parameters** » in the scrollable « LT connected » menu enables the parameter modification of the serial link between the PC and the LT.

The point : In your OPAL configuration file you set the **LT port to different values from the default** (other transmission parameters or LT slave number other than 1). Load the configuration to the LT. You would like to keep OPAL to continue testing the inputs and outputs : modify the PC transmission parameters of OPAL according to that of the LT.

VI.6- Jbus commands

This button opens a tool window which enable the **launching of Jbus master frames** from a PC. This enables, for instance, the verification of the LT memory map, before connecting the supervisor or central system. Whether a RS232 or RS485 connection the communication logic remains the same.

The window displays :

- a Jbus command entry field,
- a data report field,
- the list of available commands ,
- the list of OPAL diagnostics counters.

The general command syntax is as follows :

<i>command</i>	<i>slave nbr</i>	<i>initial address in slave</i>	<i>number of data to transfer</i>
----------------	------------------	---------------------------------	-----------------------------------

Example : Read 8 words as from address A0 in the LT slave 2 on this port: **LMOT 2 A0 8**

When clicking on the desired command in the available command list, the syntax for that command will appear in the command entry field. **One single command** may be executed in a **cyclic** manner or **case by case**.

The **communication status** signification is the same as the Jbus master protocol on an LT communication channel (refer to the chapter Exchange declaration on Modbus/Jbus master channel).

VI.7- Jbus command reports

NB : the parameters are always in a hexadecimal format. **List of Modbus/Jbus error codes (communication status):**

Decimal	Hex	Comment
0	0	exchanging
256	100	Exchange correct
769	301	Exception code : unknown function
770	302	Exception code: incorrect address
771	303	Exception code : data not valid
772	304	Exception code : slave not ready
773	305	Exception code : acknowledgement
774	306	Exception code : not acknowledged
775	307	Exception code : write error
776	308	Exception code : overlapping zones
896	380	Connection error
897	381	Connection warning
1024	400	Incorrect slave number
1025	401	Incorrect function code
1026	402	Incorrect length
1027	403	Incorrect sub-function code
1028	404	Incorrect address
1029	405	Incorrect data
1030	406	Incorrect frame length
1280	500	Slave absent
1281	501	CRC error
4096	1000	On transmission : frame underway
4097	1001	On transmission : broadcast error
4099	1004	On transmission : length error
4100	1005	On transmission : offset error
4101	1006	On transmission : function error
4102	1007	On transmission : sub-function error
4103	1008	On transmission : sub-function data error
4104	1009	On transmission : storage error

In bold the most frequently encountered communication status error codes.

VII- Processes Page

The LT allows the **local processing on internal data** whether accessible or not via the network. The operators and the processing functions or procedures are executed sequentially in the manner of the Ladder Diagram language. There are **no loop or rollback instructions**.

The operators and the functions or procedures generally work on the **array type arguments** which enable a repetitive processing of all the elements. Some functions possess arguments that enable the conditioning of the execution. Usage of these functions or procedures enable the precise creation of complex applications.

Verification of the program syntax entered by the operator is only effectuated when the Process page has been quit. You can only pass to another page if the processes syntax is correct.

The permanent LT cycle is as follows :

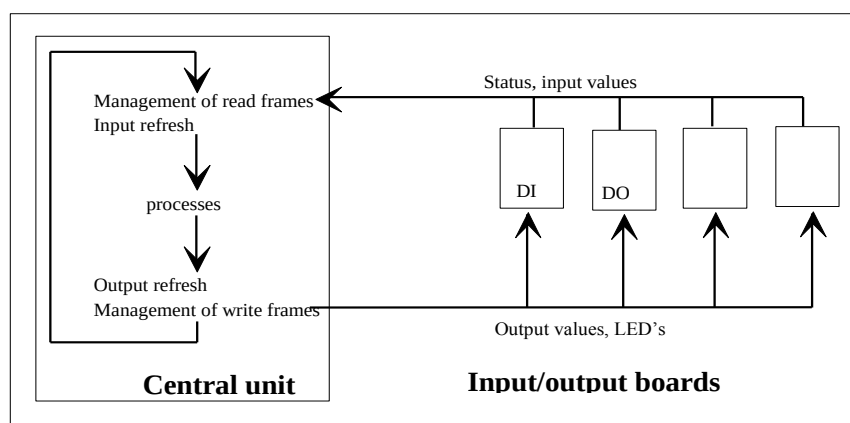


Figure 22 : LT processes cycle

If the inputs/outputs boards were not **inspected for 1 second** a monostable is triggered. The boards will therefore be in FLT mode.

VII.1- Reminder : Data definition

VII.1.1- Data types

The elementary data types are the BIT and the WORD (16 bits). A data, that is to say an entity having a name in the Data Page can also be a bit array or a word array. **In this case the data i is accessed by the expression data[i].**

Name	Type	Len	Label
STOR	BIT	16	block 3 : direct digital outputs
EANA	WORD	16	block 4 : direct analog inputs

Example according to the above data : the following operator executes the test for each analog input i (from 0 to 15) compared to 1000 and copies the test Boolean into the corresponding i output. The recursion « For i from 0 to 15 » is therefore realised implicitly.

Processes page :

STor = EAna > 1000;

The available data are those of the **"List of all LT data"** page. After setting the block and their function parameters you can rename some existing data or create new data (refer to the « Data Pages » Chapter).

VII.1.2- Internal data addresses : Alignment

The internal LT memory map is split into words. All data types, word or multiple of 16 bits are aligned according to the memory division. We would note X.Y the order bit Y in the address word X (Y between 0 and F). Example :

Address word	Order bit	Name	Type	Len.	Comment
0012	0	STOR	BIT	16	This bit variable fills the space of an entire word
0013	0	Defaults	BIT	12	This variable does not fill a whole word.
0014	0	EANA	WORD	16	The "EANA" variable is a WORD type variable. The variable "Defaults" occupies the addresses 13.0 to 13.B. the LT adds the bits 13.C to 13.F (undetermined values) to "align" the WORD to the "EANA" variable.
0024	0	Measures	WORD	3	
0027	0	Commands	BIT	5	This variable does not « fill » a complete word (27.0 to 27.4)
0027	5	Alarms	BIT	13	The "Alarms" variable is a BIT type variable which follows the "Commands » variable (27.5 to 28.1). It does not allow the « filling » of the whole word. The LT adds the bits from 28.2 to 28.F (initialised at 0) to "align" the WORD address to the « Measures » variable
0029	0	Measures	WORD	4	

The alignments are important in order to effectuate the transformation of types, Bits <> Mots variables.

VII.2. Writing a process

VII.2.1- Expression comment and terminator

Any string of characters between /* and */ is considered as a comment.

Any string between // and the end of the line is considered as a comment.

Every expression must be terminated with a semicolon ;

Numerous expressions may be written on the same line.

VII.2.2- The operators

The list of useable operators in the OPAL processes is as follows :

- **Arithmetic operators:**
 - "+" addition,
 - "-" subtraction,
 - "*" multiplication,
 - "/" division integer.
- **Binary operators:**
 - "&" AND binary,
 - "|" OR inclusive binary,
 - "^" OR exclusive binary,
 - "~" complement to 1.
- **Comparison operators :**
 - "==" strict equality,
 - "!=" unequal,
 - "<" strictly inferior,
 - ">" strictly superior.

- **Assignment operators :**
 - "=" simple assignment,
- **Auto incrementation and decrementation operators :**
Uniquely used within an expression
 - "x++" post-incrementation,
 - "++x" pre-incrementation,
 - "x--" post-decrementation,
 - "--x" pre-decrementation.
- **Various operators:**
 - "\$" concatenation of bit tables.
- **Enter a number in hexadecimal form :**
 - "Var = \$A3F",
 - or "Var = 0xA3F".

Examples : assuming that the following data are declared and named in the Data Page:

Name	Type	Size	Label
DI1	Bits	16	16 bit table of the direct inputs of block 1 DI310
DI2	Bits	16	16 bit table of the direct inputs of block 2 DI310
DO3	Bits	16	16 bit table of the direct outputs of block 3 DO310
AI4	Words	16	16 word table of the direct inputs of block 4 AI310
TB	Bit	1	User created buffer bit
TM	Word	1	User created buffer word
TM1	Word	1	User created buffer word
TM2	Word	1	User created buffer word

In the Processes Page, here are some operator usage examples:

```

DO3=DI1;           //copies block 1 to block 3: 16 input bits to the 16 output bits
DO3[7]=DI1[0];     // copies the 0 bit of block 1 to bit 7 of block 3

DO3=DI1 & DI2;      // execute an AND logic bit by bit of block 1 with block 2 and copies bit by bit
to block 3

DO3=DI1 | DI2 ;     // execute an OR logic bit by bit of block 1 with block 2 and copies bit by bit to
// block 3

DO3=DI1 ^ DI2 ;     // execute an exclusive OR logic bit by bit of block 1 with block 2 and copies bit
// by bit to block 3

DO3= ~DI1 ;         //Inverses each block bit and copies bit by bit to block 3

/* copies block 1 to block 3 by setting the bits 2 to 13 to 0*/
DO3 = DI1 & {1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1 } ;

/* copies block 1 to block 3 by setting the bits 2 to 13 to 0 and the bit 7 to 1 (pay attention to the parentheses)
*/
DO3 = ( DI1 & { 1,1,0,0,0,0,0,0,0,0,0,0,0,1,1 } ) | { 0,0,0,0,0,0,0,1,0,0,0,0,0,0,0 } ;

/* this process is also possible using the instructions :*/
DO3=DI1; DO3[2..13]=0; DO3[7]=1;

```

```

/* expansion of bits (or multiple copy) */
// in the following example the bit DI2[7] is « expanded » to the size of necessary table for the operation
// The following operation is the equivalent of the algorithm : If the bit DI2[7]=0 Then DO3=0 Else DO3=DI1
DO3=DI1 & DI2[7];

/* multiple copies */
DO3 [3..9] = DI1[8] ;      /* the DI1[8] bit is copied to the bits 3 to 9 of DO3 */

/* concatenation of table bits */
DO3 = DI1[0..7] $ DI2[8..15] ;      // copies the bits 0 to 7 of block 1 followed by the bits 8 to 15 of block
                                     // 2 to block 3
DO3 = DI2[0..13] $ {1,1} ;          // copies the bits 0 to 3 of block 1 followed by two bits at 1 on block 3

/* shift left 16 bits undertaken by copy */
DO3[0..15] = DO3[1..15] $ {0} ;

/* comparison of a table with N words to a value : result in an array of N bits */
/* Algorithm equivalent to the instruction below : */
/* For i=0 to 15 If AI4[i] >1000 Then DO3[i]=1 Else = 0 End if i Next */
DO3=AI4>1000 ;

/* comparison with Boolean processing */
DO3=(AI4<100) | (AI4>200) ;          // threshold process example

/* Fill the table with 4 values with the constants */
TAB={100 , 200 , 300 , 400 } ; // 100 in TAB[0] ,..., 400 in TAB [3]

/* additions , subtractions, multiplication's, integer divisions */
AI4[2]=AI4[0] + AI4[1] ; AI4[3]=AI4[0] - AI4[1] ;
AI4[5] = - AI4 [6];          //arithmetic opposite/
TM=TM1*TM2;                  // beware of 65535 overflow
TM = TM1/TM2;                 //beware of integer rounding

/* pre and post incrementations/decrementations */
X=++Y;                        // initially increments Y and then transfers the value to X
X=Y++;                         // transfers the value Y to X then increments X
X=--Y;                         // same with decrementation
X= Y--;                         // same with decrementation

```

VII.2.3- Typecasting

Typecasting enables the **temporary changing of a type or size of a group of contiguous variables**.

The general syntax is : BIT or WORD [length] (Variable)

This expression may be used as an argument within a function.

Constraints of transforming Words to Bits : the destination bits have to be in the addressable bit field . If this is not the case then OPAL will not accept the typecast.

```

TB1=BIT [32] (W1);           /* undertakes the typecasting of 2 consecutive W1 , W2 of the data page
                              and writes the result in binary in the 32 bits long table variable TB1 */
TB2=BIT [48] (TW[2]);        /* undertakes the typecasting of 3 consecutive words as from word 2 of the
                              TW table and writes the result as binary in the 48 bit long table variable
                              TB2 */
TB3=BIT [8] (B1);            /* undertakes the typecasting of 8 consecutive bits as from B1 and writes
                              the result as binary in the 8 bit long table variable TB3 */

```

```

TB4=BIT [16] (STOR[12]);     /* undertakes the typecasting of 16 bits as from bit 12 of a 32 bit table and
                              writes the result as binary in the 16 bit long table variable TB4 */

```

Constraints of transforming Bits to Words : the number of bits has to be a multiple of 16. The address of the 1st bit « aligned » to that of a word.

```

W1=WORD (TB16);          /* undertakes the typecasting of a 16 bit table and writes the result as
                           decimal in the W1 word variable */
W2=WORD (TB32[16..31]);  /* allows considering the 16 last bits of table TB32 as a word because
                           these last 16 bits are aligned to a word address */
                           /* undertakes the typecasting of a 16 bit table and writes the result as
                           decimal in the W2 word variable */
W3=WORD (TB32[5..20]);    /* PROHIBITED because the address of bit 5 of TB32 is not aligned with
                           a word address :a table with intermediate aligned bits must be passed :
                           refer to below */
Temp = TB32[5..20];      /* previously declare and copy to a temporary 16 bit variable */
..W4=WORD (Temp)...;

```

VII.2.4- Control instruction

_ENDINIT;

_ENDINIT is an instruction that defines a **portion of processing to be done once**. It is possible to place numerous _ENDINIT's at the start of processes. The **cyclic processes start with the process following the _ENDINIT**. If absent, all processes are executed cyclically. Example :

```

/* 1st cycle process */
Counter = 10000;
_ENDINIT;

/* Process executed in 2nd cycle */
Alarm = 1;
_ENDINIT;

/* Process executed during the 3rd cycle and the following cycles */
Counter--;
....

```

NB : the Modbus/Jbus tables aren't accessible until after the first cycle undertaken following the last _ENDINIT.

NB : the LT initialises all variables at 0 apart from, of course, the backed up data. It is therefore pointless to redo it.

VII.2.5- Conditional loop

IF <expression> THEN <expression 1> ENDIF;

IF <expression> THEN <expression 1> ELSE <expression 2> ENDIF;

Processes Page examples :

```

IF DI1[7] == 1 THEN
    DO3 = AI4 < 1000;
ELSE
    DO3 = AI4 < 2000;
ENDIF;

```

```
IF (DI1[9]==1) & (DI1[10] == 0) THEN TAB = AI4 + 500; ENDIF
```

```
IF (DI1[9]==1) & (DI1[10] == 0) THEN TAB = AI4 + 500; ENDIF
```

VII.3- The functions

VII.3.1- General syntax

destination = _FUNCTION (argument1, argument2, reference*);

ACTION	What the function does
PARAMETERS	Function input parameters
RETURN VALUE	
DESCRIPTION	If necessary the details of the algorithm executed by the function
EXAMPLE	A simple OPAL process

The function names start with an underscore.

A function always returns a value.

(*) reference is a facultative argument used in timing functions. See the corresponding paragraph.

The line must always end with a semicolon. The arguments and destinations may be arrays.

Important : Execution principle of a function :

argument1, argument2 and destination are data with n bits, words or unique elements (1 or n bits, 1 or n mots). In this case the **algorithm of any function using** this data will be:

For i from (n-1) to 0

 Execute *_FUNCTION* between **element i** of *argument1* data and element i of *argument 2* data

 Place the **result** in the *destination* element i

Next i

If argument2 is 1 word, this word is used n times for the execution of the function. This "**expansion principle**" is applicable to each parameter of a function even if it concerns a constant.

argument1, argument2 and destination may have different types (Bit or Word) according to the type of function used.

The data type for each parameter is given in the description of each function. It is the users needs which determines the size thereof.

Declaration of the data used in the examples of the following pages :

Name	Type	Size	Label
BI1	Bit	1	1 bit
MO1	Word	1	1 word
TBI1	Bit	16	16 bit array
TBI2	Bit	16	16 bit array
TBICOND	Bit	16	16 bit array
TMO1	Word	16	16 word array
TMO2	Word	16	16 word array
TMOCOND	Word	16	16 word array
TB	Bit	1	User created bit buffer
TM	Word	1	User created word buffer
TM1	Word	1	User created word buffer
TM2	Word	1	User created word buffer

VII.3.2- List of available functions

<u>_REDB</u>	Conditional detection of rising edge bits.
<u>_MEDB</u>	Conditional detection of rising and falling edge bits.
<u>_MANDB</u>	AND bit synthesis.
<u>_MORB</u>	OR bit synthesis.
<u>_MXORB</u>	XOR bit synthesis.
<u>_MEQNB</u>	Equality bit synthesis.
<u>_REC</u>	Rising edge bit count.
<u>_MEC</u>	Rising and falling edge bit count.
<u>_MIN</u>	Choose a minimum value between 2 operands.
<u>_MAX</u>	Choose a maximum value between 2 operands.
<u>_ABS</u>	Absolute word value.
<u>_SAT</u>	Word limiting between 2 thresholds.
<u>_SCAL</u>	Scaling of clipped words.
<u>_HYSB</u>	Hysteresis of words defined by 2 thresholds.
<u>_CHR</u>	Time the duration at 1 of bits.
<u>_CHRI</u>	Time the duration at 1 of bits (with rising edge reset options).
<u>_CHRC</u>	Time the duration at 1 of bits (cumulating duration at state 1).
<u>_REF</u>	Filtering of rising edge bits.
<u>_MEF</u>	Filtering of rising and falling edge bits.
<u>_DEF</u>	Filtering of rising and falling edge bits (different filter values)
<u>_BLK</u>	Generation of a square signal.
<u>_RBLK</u>	Generation of a rectangular signal.
<u>_MONB</u>	Generation of a monostable signal.
<u>_BDT</u>	Generation of an impulse signal.
<u>_CMPEH</u>	Comparison of equality between 2 double words (adapted to time management).
<u>_CMPLH</u>	Comparison of inferiority between 2 double words (adapted to time management).
<u>_CMPLH</u>	Comparison of superiority between 2 double words (adapted to time management).
<u>_CMPIH</u>	Comparison of relationship to 1 interval of 2 double words (adapted to time management).

VII.3.3- Edge detection of source bits

BitDestination = _REDB(BitSource, BitCondition);

ACTION	Conditional detection of the rising edge of the source bit and copy to the destination bit.
PARAMETERS	BitSource : data with 1 or n bits BitCondition : data with 1 or n bits
RETURN VALUE	BitDestination : data with 1 or n bits 1 or 0 in the destination bit i
DESCRIPTION	If BitCondition[i]=1 Then If Rising Edge of BitSource[i] Then BitDestination[i]=1 Else BitDestination[i]=0 Else nothing. The above algorithm implies that the destination bit remains at 1 for 1 cycle.
EXAMPLE	TBI1 = _REDB(TBI2, TBICOND);

Just inverse the source to detect the **falling edges**: ~BitSource

BitDestination = _MEDB(BitSource, BitCondition);

ACTION	Conditional detection of the rising AND falling edge of the source bit and copy to the destination bit.
PARAMETERS	BitSource : data with 1 or n bits BitCondition : data with 1 or n bits
RETURN VALUE	BitDestination : data with 1 or n bits 1 or 0 in the destination bit i
DESCRIPTION	If the BitCondition[i]=1 Then If (Rising OR Falling edge) of BitSource[i] Then BitDestination[i]=1 Else BitDestination[i]=0 Else nothing. The above algorithm implies that the destination bit remains at 1 for 1 cycle.
EXAMPLE	TBI1 = _REDB(TBI2, TBICOND);

Attention : the destination bit remains at 1 for 1 cycle. In order to extend the duration of the passage to a state 1 of the destination bit above that of the PLC cycle consists of using the monostable function (Chapter VII.3.9, page 68)

VII.3.4- Boolean synthesis with authorisations (Mask)

The following functions undertake a logical operation on the source bits to give a destination **synthesis bit**. Furthermore, the operation is effectuated on the authorised source bits according to the following principle; the destination is initialised with the neutral operation element. The mask 1 authorises the accounting of the source bit.

Description of the `_MANDB` function:

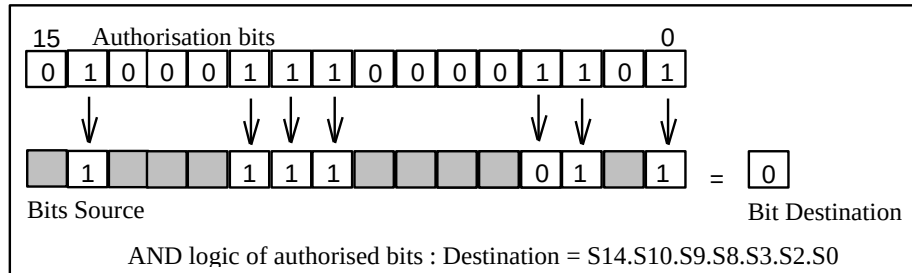


Figure 23 : Description of the execution principle of the `_MANDB` function

`BitDestination = _MANDB (BitSource, Authorisation);`

ACTION	Realise the AND synthesis of the authorised bits of the sources bits array.
PARAMETERS	BitSource : data with 1 or n bits Authorisation : data with 1 or n bits
RETURN VALUE	BitDestination : 1 bit unique, in a 1 state when the authorisation bit is not at 1.
DESCRIPTION	Refer to figure 23
EXAMPLE	BI1 = <code>_MANDB(TBI2, TBICOND);</code>

`BitDestination = _MORB (BitSource, Authorisation);`

ACTION	Realises the OR synthesis of the authorised bits.
PARAMETERS	BitSource : data with 1 or n bits Authorisation : data with 1 or n bits
RETURN VALUE	BitDestination : 1 bit unique, at 0 state when no authorisation bit is at 1. 1 or 0
DESCRIPTION	Refer to figure 23
EXAMPLE	BI1 = <code>_MORB(TBI2, TBICOND);</code>

BitDestination = _MXORB (BitSource, Authorisation);

ACTION	Realises the EXCLUSIVE OR synthesis of authorised bits.
PARAMETERS	BitSource : data with 1 or n bits authorisation : data with 1 or n bits
RETURN VALUE	BitDestination : 1 bit unique, in a 0 state when no authorisation bits are at 1. 1 or 0
DESCRIPTION	Refer to figure 23
EXAMPLE	BI1 = _MXORB(TBI2, TBICONDD); The destination bit is at 1 (respectively 0) if the number of source bits in a 1 state is ODD (respectively EVEN)

BitDestination = _MEQNB (BitSource, authorisation);

ACTION	Realises the equivalence synthesis of the authorised bits.
PARAMETERS	BitSource : data with 1 or n bits Authorisation : data with 1 or n bits
RETURN VALUE	BitDestination : 1 bit unique, initialised by the neutral 0 element. 1 or 0
DESCRIPTION	Refer to figure 23
EXAMPLE	BI1 = _MEQNB(TBI2, TBICONDD); The destination bit is at 1 (respectively at 0) if the number of authorised source bits in a 0 state is EVEN (respectively ODD)

VII.3.5- Digital functions

The manipulated integers (words) are **signed integers**.

WordDestination = _REC(BitSource, BitCondition);	
ACTION	Conditional count of the number of rising edge source table bits.
PARAMETRES	BitSource : data with 1 or n bits BitCondition : data with 1 or n bits
RETURN VALUE	WordDestination : 1 or n words data A counter word for each source table bit
DESCRIPTION	If (BitCondition[i]=1 AND rising edge of BitSource[i]) Then Integer destination[i]++; Endif;
EXAMPLE	TMO1 = _REC(TBI2, TBICOND);

WordDestination = _MEC (BitSource, Bit Condition);	
ACTION	Conditional count of the number of rising and falling edge source table bits.
PARAMETERS	BitSource : data with 1 or n bits BitCondition : data with 1 or n bits
RETURN VALUE	WordDestination : 1 or n word data A counter word for each source table bit
DESCRIPTION	If BitCondition[i]=1 AND (rising OR descending edge) of BitSource[i] Then Integer destination[i]++; Endif;
EXEMPLE	TMO1 = _MEC(TBI2, TBICOND);

WordDestination = _MIN(Operand1, Operand2);	
ACTION	Choose the minimum value between 2 operands.
PARAMETERS	Operand1 : 1 or n words data Operand2 : 1 or n words data
RETURN VALUE	WordDestination : 1 or n words data
DESCRIPTION	WordDestination[i] = minimum(Operand1[i], Operand2[i]);
EXAMPLE	TMORES = _MIN(TMO1, TMO2);

WordDestination = _MAX(Operand1, Operand2);

ACTION	Choose the maximum value between 2 operands.
PARAMETERS	Operand1 : 1 or n words data Operand2 : 1 or n words data
RETURN VALUE	WordDestination : 1 or n words data
DESCRIPTION	WordDestination[i] = maximum(Operand1[i], Operand2[i]);
EXAMPLE	TMORES = _MAX(TMO1, TMO2);

WordDestination = _ABS(WordSource , BitCondition);

ACTION	absolute value.
PARAMETERS	WordSource : data with 1 or n words BitCondition : data with 1 or n bits
RETURN VALUE	WordDestination : data with 1 or n words
DESCRIPTION	If (BitCondition[i]=1) then WordDestination[i] = absolute value (WordSource); Endif;
EXAMPLE	TMORES = _ABS(TMO1, TBICOND);

WordDestination = _SAT(WordSource, thresholds);

ACTION	Clipping each word of the WordSource table words between 2 thresholds.
PARAMETERS	WordSource : data with 1 or n words Thresholds : word table with lower threshold = Threshold[i,0] and upper threshold = threshold[i,1] (1 element = 2 words)
RETURN VALUE	WordDestination : data with 1 or n words
DESCRIPTION	If (WordSource[i] < Threshold[i,0]) Then WordDestination[i] = Threshold[i,0]; Else WordDestination[i] = WordSource[i]; Endif; If (WordSource[i] > threshold[i,1]) Then // WordDestination updated above WordDestination[i] = Threshold[i,1]; Else WordDestination[i] = WordSource[i]; Endif; No verification is undertaken on the upper threshold. As a consequence if Threshold [1] < Threshold [0], the result would be equal to Threshold [1] in all cases
EXAMPLE	TMORES = _SAT(TMO1, TTHRESHOLDS);

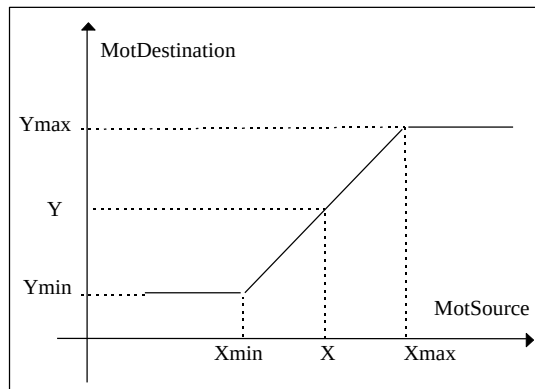
WordDestination = _SCAL(WordSource, PointsMech);

ACTION Scaling with clipping.

PARAMETERS WordSource : data with 1 or n words
 PointsMech : n word data with :
 Xmin == PointsMech[i,0];
 Xmax == PointsMech[i,1];
 Ymin == PointsMech[i,2];
 Ymax == PointsMech[i,3];
 (1 element = 4 words)

RETURN VALUE WordDestination : data with 1 or n words

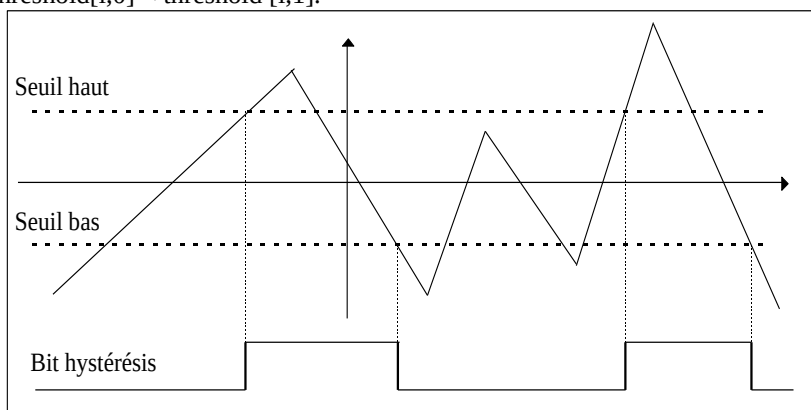
DESCRIPTION Each couple (WordSource, WordDestination) is associated with a transformation line defined by 2 points (Xmin, Ymin) and (Xmax, Ymax) according to the graph below. If the source is outside the Xmin Xmax interval the destination is clipped at Ymin or Ymax.



EXAMPLE TMORES = _SAT(TMO1, TPMECH);

BitDestination = _HYSB (WordSource, Thresholds);

ACTION	The destination bit follows the hysteresis trace defined by the 2 thresholds.
PARAMETERS	WordSource : data with 1 or n words Thresholds : Thresholds : word table with lower threshold = Threshold[i,0] and upper threshold = threshold[i,1] (1 element = 2 words)
RETURN VALUE	BitDestination : data with 1 or n bits
DESCRIPTION	The 2 tests are undertaken in the following order : If WordSource[i] < threshold[i,0] Then BitDestination[i] = 0; Endif ; If WordSource[i] > threshold [i,1] Then BitDestination[i] = 1; Endif ; NB : To ensure the correct operation you must of course watch that threshold[i,0] < threshold [i,1].



EXAMPLE TBIRES = _HYSB(TMO1, TTHRESHOLDS);

VII.3.6- Timer functions

Timer functions introduce a **supplementary parameter**: the **reference timer** for counting time. All of the timer functions are based on the reference timer. That is to say a counter incremented at each period by the basic time. The timers are based on the LT equipment timer which counts in milliseconds and whose address is arbitrarily 0. The integers (preset, counter, etc.) of the timer functions are **unsigned integers**.

Timer = _CHR(BitStartStop ,IntegerPreset , ReferenceTimer);	
ACTION	Times the duration at 1 of BitStartStop. The BitStartStop rising edge starts the timer without reinitialising it . The BitStartStop falling edge stops the timer.
PARAMETERS	BitStartStop : data with 1 or n bits IntegerPreset : data with 1 or n words (milliseconds) ReferenceTimer : addr(0) (milliseconds)
RETURN VALUE	Timer : data with 1 or n words
DESCRIPTION	If (BitStartStop[i]=1) Then Integer Timer[i]++; when time passed equal to IntegerPreset[i] with unit given by ReferenceTimer. Endif; If Integer Preset[i]=0 Then _CHR is without effect. ReferenceTimer may in fact be any counter : Timer will be incremented by 1 for each quantity equal to IntegerPreset x ReferenceTimer
EXAMPLE	Time in seconds the duration at 1 of each digital input of a DI310 board: Digital Inputs : 32 bits (direct inputs from a DI310) Duration's : 32 words (duration in seconds in a 1 state) <i>Duration = _CHR(DigitalInputs, 1000, addr(0));</i> 1000 is the timer preset: 1000 milliseconds = 1s. The Durations[i] values are reset to 0 at each DigitalInputs[i] rising edge and augmented by 1 at each second if DigitalInputs[i] =1. In this example it is not possible to time in hours because the preset would exceed 65535 milliseconds. It is therefore necessary to create intermediate reference timers based on the millisecond counters of the equipment (refer to Other example).

Comment : on _CHR restart the memory if the reference units are deleted.

Illustration : That is the following _CHR values at the time of stoppage (passage from 1 to 0 of BitStartStop) :

- IntegerPreset = 1000
- Timer = 45
- ReferenceTimer (invisible) = 800, there were still 200 reference units left before the Timer passed the 46 value.

At restart (following passage from 0 to 1 of BitStartStop), the reference unit counter is reset to 0. The Timer restarts as from 45 but 800 reference units have been lost.

Use the _CHRC function if you do not want this functionality.

Other example : Timer cascades

We would like to time the running times of the motors in hours. The millisecond timer preset being too large it is necessary to create an intermediate « cascade » timer.

MotorsInputs : 16 bits (Status of the 16 first inputs of the DI310 board)

Second : 1 word (Seconds timer (created))

HoursRunning : 16 words (Timers of running hours (in hours))

```
/* creating the intermediate reference timer */
```

```
Second = _CHR(1, 1000, addr(0));
```

Second : timer result incremented permanently by 1 every 1000 milliseconds.

1 : the (start/stop bit) forced to 1 permanently.

1000 : timing preset : 1000 milliseconds = 1s.

addr(0) : address of the equipment millisecond timer.

```
/* Timing of running hours*/
```

```
HoursRunning = _CHRC(MotorsInputs, 3600, Second);
```

The Second timer, created above, is used as the reference timer.

HoursRunning is incremented by 1 every 3600 x Second if MotorsInputs =1.

```
Timer = _CHRI(BitStartStop, IntegerPreset, ReferenceTimer);
```

ACTION	Time the duration at 1 of BitStartStop. The BitStartStop rising edge starts the timer by reinitialising it . The BitStartStop falling edge stops the.
PARAMETERS	BitStartStop : data with 1 or n bits IntegerPreset : data with 1 or n words (milliseconds) ReferenceTimer : addr(0) (milliseconds)
RETURN VALUE	Timer : data with 1 or n words
DESCRIPTION	Same _CHR
EXAMPLE	Same _CHR

```
Timer = _CHRC(BitStartStop, Integer Preset, ReferenceTimer);
```

ACTION	Time the duration at 1 of BitStartStop. The BitStartStop rising edge starts the timer without initialisation . The BitStartStop falling edge stops the timer.
PARAMETERS	BitStartStop : data with 1 or n bits IntegerPreset : data with 1 or n words (milliseconds) ReferenceTimer: addr(0) (milliseconds)
RETURN VALUE	Timer : data with 1 or n words
DESCRIPTION	Same as _CHR but CHRC does not delete the counter reference units on startup. (refer to CHR paragraph).
EXAMPLE	Same as _CHR

Synthesis of time functions in diagram form :

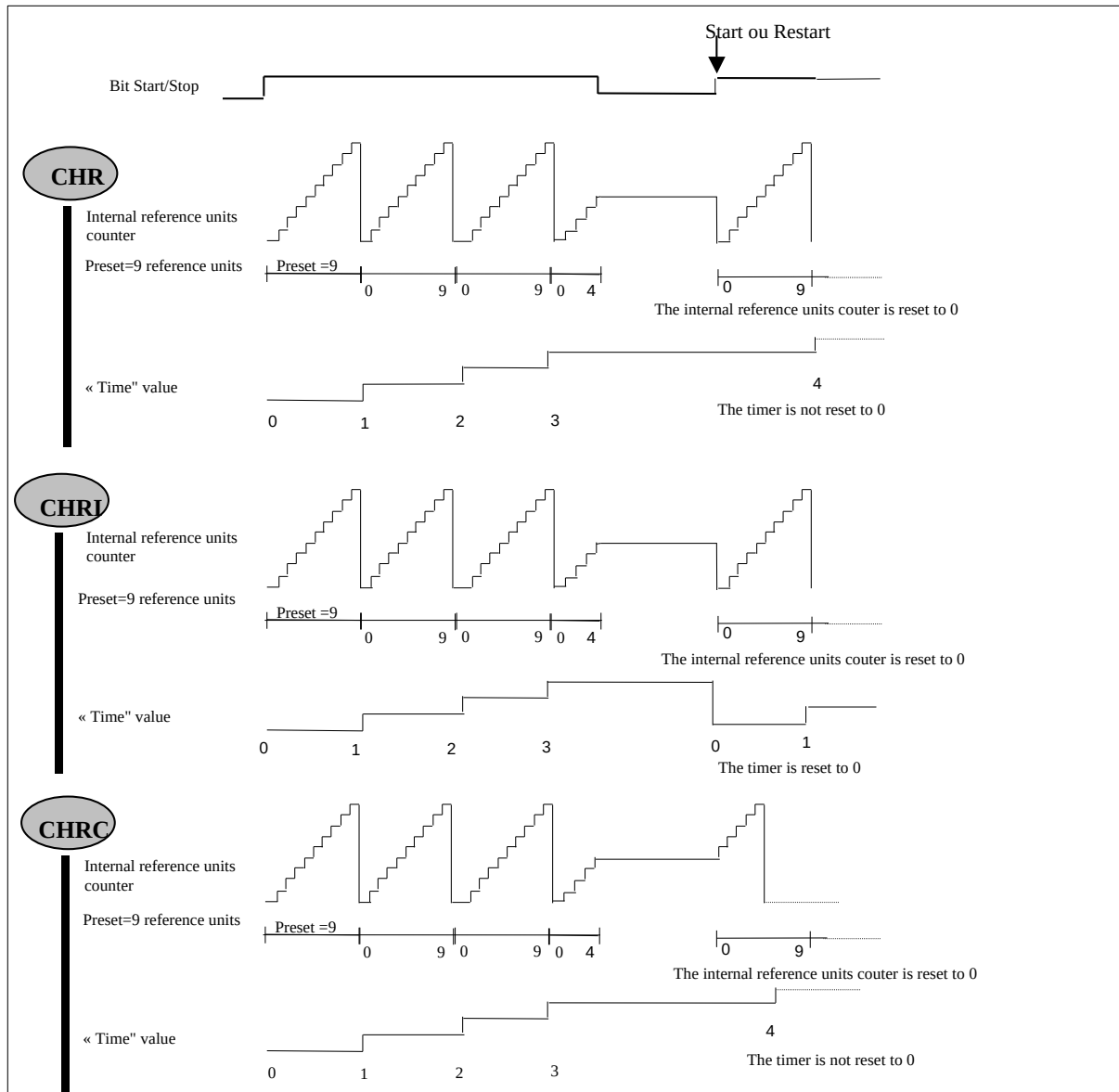


Figure 24 : Synthesis of time functions

VII.3.7- Edge filters

BitDestination = _REF(BitSource, IntegerPreset, ReferenceTimer);

ACTION	Filter on rising edge of bit.
PARAMETERS	BitSource : data with 1 or n bits IntegerPreset : data with 1 or n words (milliseconds) ReferenceTimer: addr(0) (milliseconds)
RETURN VALUE	BitDestination : data with 1 or n bits
DESCRIPTION	BitDestination passes to 1 if the BitSource is at 1 for at least IntegerPreset x ReferenceTimer.
EXAMPLE	// Filter the rising edges of 16 words with IntegerPreset = 2000 milliseconds TMO1 = _REF (TBI1, 2000, addr(0));

BitDestination = _MEF(BitSource, IntegerPreset, ReferenceTimer);

ACTION	Filtering the rising AND falling edge of a bit
PARAMETERS	BitSource : data with 1 or n bits IntegerPreset : data with 1 or n words (milliseconds) ReferenceTimer: addr(0) (milliseconds)
RETURN VALUE	BitDestination : data with 1 or n bits
DESCRIPTION	The BitDestination passes to 1 (respectively to 0) if BitSource is at 1 (respectively 0) for at least IntegerPreset x ReferenceTimer. The preset value is the same for the rising and falling edge.
EXAMPLE	// Filter the rising and falling edges of 16 words with IntegerPreset = // 1000 milliseconds TMO1 = _MEF (TBI1, 1000, addr(0));

Destination = _DEF (BitSource, IntegerPreset, ReferenceTimer);

ACTION	Filter the rising And falling edge of a bit
PARAMETERS	BitSource : data with 1 or n bits IntegerPreset : data with 1 or n words (milliseconds) (1 element = 2 words) ReferenceTimer: addr(0) (milliseconds)
RETURN VALUE	BitDestination : data with 1 or n bits
DESCRIPTION	Same as _MEF but the filtering duration's are different for the rising edge and falling. "Preset" is a 2 column array with : - IntegerPreset[0] = Preset for the filter duration of the falling edge - IntegerPreset[1] = Preset for the filter duration of the rising edge
EXAMPLE	// Filter the rising and falling edges of 16 words with the following values: 1000, 2000 milliseconds FILTERS = {1000, 2000}; _ENDINIT; TMO1 = _MEF (TBI1, FILTERS, addr(0));

VII.3.8- Square and rectangular signals

Bit Destination = _BLK (BitSource, IntegerPreset, ReferenceTimer);

ACTION	Generate an authorised square signal.
PARAMETERS	BitSource : data with 1 or n bits IntegerPreset : data with 1 or n words (milliseconds) ReferenceTimer: addr(0) (milliseconds)
RETURN VALUE	BitDestination : data with 1 or n bits
DESCRIPTION	If BitSource =1 Then BitDestination is a square signal whose duration at 1 and 0 is given by IntegerPreset x ReferenceTimer; Else BitDestination = 0;
EXAMPLE	TBIRES = _BLK (TBI1, 2000, addr(0));

BitDestination = _RBLK (BitSource, IntegerPreset, ReferenceTimer);

ACTION	Generate an authorised rectangular signal
PARAMETERS	BitSource : data with 1 or n bits IntegerPreset : data with 1 or n words (milliseconds) (1 element = 2 words) ReferenceTimer: addr(0) (milliseconds)
RETURN VALUE	BitDestination : data with 1 or n bits
DESCRIPTION	If BitSource =1 Then BitDestination is a rectangular signal whose 0 duration is given by IntegerPreset[0] x ReferenceTimer the 1 duration given by IntegerPreset[1] x ReferenceTimer; Else BitDestination = 0;
EXAMPLE	TBIRES = _RBLK (TBI1, INTEGERPRESET, addr(0));

VII.3.9- Monostable

BitDestination = _MONB(BitSource, IntegerPreset, ReferenceTimer);

ACTION	Generate a Monostable signal triggered on a rising edge of the source bit.
PARAMETERS	BitSource : data with 1 or n bits IntegerPreset : data with 1 or n words (milliseconds) ReferenceTimer: addr(0) (milliseconds)
RETURN VALUE	BitDestination : data with 1 or n bits
DESCRIPTION	The IntegerPreset monostable duration is re-launched at each rising edge du of the source bit even if the duration has not been fully used.
EXAMPLE	TBIRES = _MONB(TBI1, 20, addr(0));

VII.3.10- Impulse

```
BitDestination = _BDT(BitSource, IntegerPreset, ReferenceTimer);
```

ACTION	Generate an impulse bit.
PARAMETERS	BitSource : data with 1 or n bits IntegerPreset : data with 1 or n words (milliseconds) ReferenceTimer: addr(0) (milliseconds)
RETURN VALUE	BitDestination : data with 1 or n bits
DESCRIPTION	Generates an impulse BitDestination at IntegerPreset x ReferenceTimer after a rising edge BitSource. This bit remains at 1 during 1 cycle. If no read has occurred immediately after the impulse rise, ReferenceTimer continues to count the duration of the following re-launch.
EXAMPLE	TBIRES = _BDT(TBI1, 2000, addr(0));

VII.3.11- Comparisons of double words (comparing dates and times)

Comment : although the functions are adapted to date comparisons they are not specifically for that : they compare 2 double words by rising addressing bytes. The words used are **signed integers**.

The LT clock can be inserted into the public data list in the **Parameters Page**. The clock is composed of 4 words according to the following :

Order	High byte	Low byte
0	seconds	1/100th of a sec
1	hours	minutes
2	Day of month	Day of week [0(Monday) ..6(Sunday)]
3	year	month

BitDestination = _CMPEH(Operand1, Operand2);

ACTION	Equality comparison of two double words.
PARAMETERS	Operand1 : n word data (1 element = 2 words) Operand2 : : n word data (1 element = 2 words)
RETURN VALUE	BitDestination : data with 1 or n bits
DESCRIPTION	Byte by byte equality comparison between Operand1 double word and the Operand 2 double word. The result is a bit at 1 if equality and 0 if not.
EXAMPLE	Compare the current date available in the CPU time stamp with 2 holiday constant in 1996: New Years day and Christmas day and place a « Holiday » bit <pre>// HolDates : 4 words // HolDays : 2 bits // Holiday : 1 bit // Monday 1/1/96 and Wednesday 25/12/96 : HolDates = {0x0100, 0x6001, 0x1902, 0x600C}; _ENDINIT; HolDays = _CMPEH(HolDates, TimeStamp[2..3]); Holiday = _MORB(HolDays, 1); // OR synthesis of the HolDays array</pre>

BitDestination = _CMPLH(Operand1, Operand2);

ACTION	Inferiority comparison of two double words.
PARAMETERS	Operand1 : n word data (1 element = 2 words) Operand2 : n word data (1 element = 2 words)
RETURN VALUE	BitDestination : data with 1 or n bits
DESCRIPTION	Comparison of inferiority byte by byte of the Operand1 double word and Operand2 double word. The result is a bit at 1 if Operand1 is less than Operand2 else at 0.
EXAMPLE	Same as _CEMPH

BitDestination = _CMPGH(Operand1, Operand2);

ACTION	Superiority comparison of two double words.
PARAMETERS	Operand1 : n word data (1 element = 2 mots) Operand2 : n word data (1 element = 2 mots)
RETURN VALUE	BitDestination : data with 1 or n bits
DESCRIPTION	Byte by byte superiority comparison between the Operand1 double word and the Operand2 double word. The result is a bit set to 1 if Operand1 is greater than Operand2 else set to 0.
EXAMPLE	Same as _CEMPH

BitDestination = _CMPIH (Operand1, Operand2);

ACTION	Double word property of an interval defined by two double words .
PARAMETERS	Operand1 : n word data (1 element = 2 words) Operand2 : n word data (1 element = 4 words)
RETURN VALUE	BitDestination : data with 1 or n bits
DESCRIPTION	Byte by byte relationship of the Operand1 double word to an interval defined by the Operand2[0..1] double word and the Operand2[2..3] double word. The result is a bit at 1 if Operand1 is within the two double words else at 0.
EXAMPLE	Same as _CEMPH

VII.4- The procedures

VII.4.1- General syntax

<i>PROCEDURE (argument1, argument2, destination);</i>	
ACTION	What the procedure achieves
PARAMETERS	Parameters at procedure input
RESULT	Parameters at procedure output
DESCRIPTION	If necessary, describes the algorithm executed by the procedure
EXAMPLE	Simple OPAL process example.

Procedure names start with an underscore.

A procedure does not return a value. The result is in the destination argument.

If the execution conditions are not correct then the result is not placed in the destination.

The line must be terminated with a semicolon. The arguments and the destination may be arrays.

Important : Execution principle of a procedure :

argument1, argument2 and destination are data with n bits, words or unique elements (1 or n bits, 1 or n mots). In this case the **algorithm of any procedure using** this data will be:

For i from (n-1) to 0

 Execute *_PROCEDURE* between **element i** of *argument1* data and element i of *argument 2* data

 Place the **result** in the *destination* element i

Next i

If *argument2* is 1 word, this word is used n times for the execution of the function. This "**expansion principle**" is applicable to each parameter of a procedure even if it concerns a constant.

argument1, *argument2* and *destination* may have different types (Bit or Word) according to the type of function used.

Only the data type for each parameter is given in the description of each procedure. It is the users requirements which determines the size thereof.

VII.4.2- List of available procedures

<i>_MOV</i>	Conditional transfer of bits to BitDestination according to the BitCondition condition.
<i>_MOV</i>	Conditional word transfer.
<i>_MOVD</i>	Conditional double word transfer
<i>_MOVQ</i>	Conditional quadruple word transfer.
<i>_REMB</i>	Conditional transfer of rising or falling edge bits
<i>_REM</i>	Conditional transfer of rising or falling edge words.
<i>_REMD</i>	Conditional transfer of rising or falling edge double words.
<i>_REMQ</i>	Conditional transfer of rising or falling edge quadruple words.
<i>_MEMB</i>	Conditional transfer of rising and falling edge bits
<i>_MEM</i>	Conditional transfer of rising and falling edge words.
<i>_MEMD</i>	Conditional transfer of rising and falling edge double words.
<i>_MEMQ</i>	Conditional transfer of rising and falling edge quadruple words.

VII.4.3- Transfer procedure on state of condition

<i>_MOVB(BitSource, BitCondition, BitDestination);</i>	
ACTION	Conditional transfer of the BitSource to BitDestination according to the BitCondition condition.
PARAMETERS	BitSource : data with 1 or n bits BitCondition : data with 1 or n bits
RETURN VALUE	BitDestination : data with 1 or n bits
DESCRIPTION	If BitCondition[i] = 1 Then copy BitSource[i] in BitDestination[i] Else nothing; EndIf;
EXAMPLE	_MOVB(TBI1, TBI2, TBRES);

<i>_MOV(WordSource, BitCondition, WordDestination);</i>	
ACTION	Conditional transfer of WordSource to WordDestination according to the BitCondition conditions.
PARAMETERS	WordSource : data with 1 or n words BitCondition : data with 1 or n bits
RETURN VALUE	WordDestination : data with 1 or n words
DESCRIPTION	If BitCondition[i] = 1 Then copy WordSource[i] into WordDestination[i] Else nothing; Endif;
EXAMPLE	_MOV(TMO1, TBI2, TMORES);

_MOVD(WordSource, BitCondition, WordDestination);

ACTION	Conditional transfer of WordSource to WordDestination according to the BitCondition condition. WordSource and WordDestination are double words
PARAMETERS	WordSource : n word data (1 element = 2 words) BitCondition : data with 1 or n bits
RETURN VALUE	WordDestination : n word data (1 element = 2 words)
DESCRIPTION	If BitCondition[i] = 1 Then copy WordSource[i] to WordDestination[i] Else nothing; Endif ;
EXAMPLE	_MOVD(TMO1, TBI2, TMORES);

_MOVQ(WordSource, BitCondition, WordDestination);

ACTION	Conditional transfer of WordSource to WordDestination according to the BitCondition condition. WordSource and WordDestination are quadruple words
PARAMETERS	WordSource : n word data (1 element = 4 words) BitCondition : data with 1 or n bits
RETURN VALUE	WordDestination : n word data (1 element = 4 words)
DESCRIPTION	If BitCondition[i] = 1 Then copy WordSource[i] to WordDestination[i] Else nothing; Endif ;
EXAMPLE	_MOVQ(TMO1, TBI2, TMORES);

Example : Fill a TBModbus array with analog measures of an AI210 block if these values are less than their upper thresholds else fill the array with the max. thresholds values.

Data Page :

ValThresholds	WORD	16	Threshold value array
DepThresholds	BIT	16	Thresholds overflow bits array
AI	WORD	16	16 analog inputs array
TBModbus	WORD	16	Destination array

Processes page :

```

DepThresholds = AI > ValThresholds;  threshold calculation
// transfer of measures. The condition is the inverted DepThresholds array
_MOV(AI, ~DepThresholds, TBModbus);

// transfer of threshold values. The condition is the normal DepThresholds normal
_MOV(ValThresholds, DepThresholds, TBModbus);

```

VII.4.4- Transfer procedures on the rising edge of a condition

Inverting the BitCondition: **~BitCondition**; is sufficient for achieving the transfer on the falling edge of the condition

REMB(BitSource, BitCondition, BitDestination);	
ACTION	Conditional transfer of the BitSource in BitDestination on the rising edge according to the BitCondition condition.
PARAMETERS	BitSource : data with 1 or n bits BitCondition : data with 1 or n bits
RETURN VALUE	BitDestination : data with 1 or n bits
DESCRIPTION	If (rising edge of BitCondition[i]) then copy BitSource[i] to BitDestination[i] Else nothing; Endif ;
EXAMPLE	_REMB(TBI1, TBI2, TBRES);

REM(WordSource, BitCondition, WordDestination);	
ACTION	Conditional transfer of WordSource to WordDestination on rising edge according to the BitCondition condition.
PARAMETERS	WordSource: data with 1 or n words BitCondition : data with 1 or n bits
RETURN VALUE	WordDestination: data with 1 or n words
DESCRIPTION	If (rising edge of BitCondition[i]) Then copy WordSource[i] to WordDestination[i] Else nothing; Endif ;
EXAMPLE	_REM(TMO1, TBI2, TMORES);

REMD(WordSource, BitCondition, WordDestination);

ACTION	Conditional transfer of WordSource to WordDestination on rising edge according to the BitCondition condition.. WordSource and WordDestination are double words.
PARAMETERS	WordSource: n word data (1 element = 2 words) BitCondition : data with 1 or n bits
RETURN VALUE	WordDestination: n word data (1 element = 2 words)
DESCRIPTION	If (rising edge of BitCondition[i]) Then copy WordSource[i] to WordDestination[i] Else nothing; Endif ;
EXAMPLE	_REMD(TMO1, TBI2, TMORES);

REMQ(WordSource, BitCondition, WordDestination);

ACTION	Conditional transfer of WordSource to WordDestination on rising edge according to the BitCondition condition.. WordSource and WordDestination are quadruple words.
PARAMETERS	WordSource: n word data (1 element = 4 words) BitCondition : data with 1 or n bits
RETURN VALUE	WordDestination: n word data (1 element = 4 words)
DESCRIPTION	If (rising edge of BitCondition[i]) Then copy WordSource[i] to WordDestination[i] Else nothing; Endif ;
EXAMPLE	_REMQ(TMO1, TBI2, TMORES);

VII.4.5- Transfer procedures on the rising and falling edge of a condition***MEMB(BitSource, BitCondition, BitDestination);***

ACTION	Conditional transfer of BitSource to BitDestination on rising AND falling edge according to BitCondition condition.
PARAMETERS	BitSource : data with 1 or n bits BitCondition : data with 1 or n bits
RETURN VALUE	BitDestination : data with 1 or n bits
DESCRIPTION	If (rising edge of BitCondition[i]) Then copy BitSource[i] to BitDestination[i] Else nothing; Endif ;
EXAMPLE	_MEMB(TBI1, TBI2, TBRES);

MEM(WordSource, BitCondition, WordDestination);

ACTION	Conditional transfer of WordSource to WordDestination on rising AND falling edge according to BitCondition condition
PARAMETERS	WordSource: data with 1 or n words BitCondition : data with 1 or n bits
RETURN VALUE	WordDestination: data with 1 or n words
DESCRIPTION	If (rising edge of BitCondition[i]) Then copy WordSource[i] to WordDestination[i] Else nothing; Endif ;
EXAMPLE	_MEM(TMO1, TBI2, TMORES);

MEMD(WordSource, BitCondition, WordDestination);

ACTION	Conditional transfer of WordSource to WordDestination on rising AND falling edge according to BitCondition condition. WordSource and WordDestination are double words.
PARAMETERS	WordSource: n word data (1 element = 2 words) BitCondition : data with 1 or n bits
RETURN VALUE	WordDestination: n word data (1 element = 2 words)
DESCRIPTION	If (rising edge of BitCondition[i]) Then Copy WordSource[i] to WordDestination[i] Else nothing; Endif ;
EXAMPLE	_MEMD(TMO1, TBI2, TMORES);

MEMQ(WordSource, BitCondition, WordDestination);

ACTION	Conditional transfer of WordSource to WordDestination on rising AND falling edge according to BitCondition condition. WordSource and WordDestination are quadruple words.
PARAMETERS	WordSource: n word data (1 element = 4 words) BitCondition : data with 1 or n bits
RETURN VALUE	WordDestination: n word data (1 element = 4 words)
DESCRIPTION	If (rising edge of BitCondition[i]) Then Copy WordSource[i] to WordDestination[i] Else nothing; Endif ;
EXAMPLE	_MEMQ(TMO1, TBI2, TMORES);

VIII- DIAGNOSTICS AND IMPLEMENTATION

ATTENTION : the LT100 initialisation lasts approximately 7 seconds.

VIII.1- Diagnostics of the PSD230 (LT100) power supply

Serigraphy	LED Colour	LED state	Signification
Unom (+24/48V)	green		Voltage supply present and correct
OK	green	Lit by CU	Internal voltage correct
FLT	red	lit	Incorrect internal voltages of power board not set
+24V/0.2A	green	fixed	Voltage available for inputs/outputs
Wdg	red		General system defect
Prm	orange	fixed	The bridge between 2 Prm terminal blocks is present

VIII.2- Diagnostics of the central units CPU1xx (LT100)

Serigraphy	LED Colour	LED state	Signification
Run	green	Flashing rapidly (frequency : 2/10s)	The CU executes system software and the parameterised application. The COM1.1 link is in parameter setting mode
		Flashing slowly (frequency : 2s)	The CU executes system software and the parameterised application.
Test	green	Permanently lit	The CU auto-tests are correct
		flashing mainly lit	Application defect (OPAL Processes Page) (lit for 3 periods extinguished for 1)
		flashing mainly extinguished	Defect equipment parameter setting(OPAL Equipment Page) (extinguished 3 periods, lit 1 period)
		extinguished	Defect in the saved parameters
I/O	green	Permanently lit	Inputs/outputs correct Attention: The power supply module is part of the I/O modules
		flashing mainly lit	Boards recognised but not all managed
		flashing mainly extinguished	Absent or incorrect boards according to the OPAL parameter settings
		extinguished	No I/O managed=> general internal bus defect
Prm	green	Permanently lit	Operation with default parameter settings. The client application is not executed.
Prg	green		reserved
Flt	orange	Permanently lit	Synthesis of Test and I/O defects: lit , if I/O extinguished or flashing or Test extinguished or flashing
Wdg	red	Permanently lit	The CU is not operational : in Init out of order. (may be flashing if permanent reset)
Comi (i=0,1)	green	Permanently lit	The communication port is active (A communication protocol presence)
		flashing	The port alarm is active (parameterised by OPAL
FIP	green	Permanently lit	The fulfil component is correctly initialised

VIII.3- Diagnostics of the PSD300 (LT80) - PSD301 (LT160) power supplies

Serigraphy	LED Colour	LED state	Signification
OK	green		Correct voltage

VIII.4- Diagnostics of the CPU3xx central units (LT80 and LT160)

Serigraphy	LED Colour	LED state	Signification
Run	green	flashing rapidly (frequency : 2/10s)	The CU executes system software and the parameterised application. The COM1.1 link is in parameter setting mode
		Flashing slowly (frequency : 2s)	The CU executes system software and the parameterised application.
Test	red	extinguished	The CU auto-tests are correct
		flashing mainly extinguished	Application defect (OPAL Processes Page) (lit for 3 periods extinguished for 1)
		flashing mainly lit	Defect equipment parameter setting(OPAL Equipment Page) (extinguished 3 periods, lit 1 period)
		extinguished	Defect in the saved parameters
I/O	red	extinguished	Inputs/outputs correct Attention: The power supply module is part of the I/O modules
		flashing mainly extinguished	Boards recognised but not all managed
		flashing mainly lit	Absent or incorrect boards according to the OPAL parameter settings
		extinguished	No I/O managed=> general internal bus defect
Prm	green	lit permanently	Operation with default parameter settings. The client application is not executed.
Prg	green		reserved
Wdg	red	lit permanently	The CU is not operational : an Init out of order. (may be flashing if permanent reset)
Com i i=[0..1] on LT80 i=[0..7] on LT160	green	lit permanently	The communication port is active (A communication protocol presence)
		flashing	The port alarm is active (parameterised by OPAL
FIP (LT80)	green	lit permanently	The FullFIP component is correctly initialised

VIII.5- Diagnostics of the COMxxx communication modules

Serigraphy	LED Colour	LED state	Signification
Tx	green	lit	Transmission wire activated (equipment activation)
Rx	green	lit	Reception wire activated (equipment activation)
Ten	green	lit	UNDER JBUS/MODBUS PROTOCOL : lit by software when LT. This LED corresponds to the RTS output pin 7 of the SubD; it enables the possibility of piloting external transmitters or converters.
			<i>No FIP display led</i>

VIII.6- Diagnostics Dixxx digital input modules

Serigraphy	LED Colour	LED state	Signification															
FLT	red	lit permanently	If general WDG Or if board internal power defect Or if no access via the CU software (if not parameterised, or refresh time > 1 second)															
1 LED per channel	green	lit permanently	LED lit by central unit If the corresponding input on the terminal block is seen to be 1 by the CU															
DI312 case 2 LEDs per channel	green and red		<table><tr><td><i>Led red</i></td><td><i>Led green</i></td><td><i>Signification</i></td></tr><tr><td>extinguished</td><td>extinguished</td><td>captor open</td></tr><tr><td>extinguished</td><td>lit</td><td>captor closed</td></tr><tr><td>lit</td><td>extinguished</td><td>input not connected CC at 0V</td></tr><tr><td>lit</td><td>lit</td><td>short circuit at +V</td></tr></table>	<i>Led red</i>	<i>Led green</i>	<i>Signification</i>	extinguished	extinguished	captor open	extinguished	lit	captor closed	lit	extinguished	input not connected CC at 0V	lit	lit	short circuit at +V
<i>Led red</i>	<i>Led green</i>	<i>Signification</i>																
extinguished	extinguished	captor open																
extinguished	lit	captor closed																
lit	extinguished	input not connected CC at 0V																
lit	lit	short circuit at +V																

VIII.7- Diagnostics of the Doxxx digital output modules

Serigraphy	LED Colour	LED state	Signification
FLT	red	lit permanently	If general WDG Or if board internal power defect Or if no access via the CU software (if not parameterised)
1 LED per channel	orange	lit permanently	LED lit by the central unit According to the activation of the output on the terminal block

VIII.8- Diagnostics of the DIO210 digital inputs/outputs modules

Serigraphy	LED Colour	LED state	Signification
FLT	red	lit permanently	If general WDG Or if board internal power defect Or if no access via the CU software (if not parameterised)
1 LED per channel	green	lit permanently	LED lit by central unit If the CU supports the refresh of this channel

VIII.9- Diagnostics of the AIxxx analog input modules

Serigraphy	LED Colour	LED state	Signification
FLT	red	lit permanently	If general WDG Or if board internal power defect Or if no access via the CU software (if not parameterised)
1 LED per channel	green	lit permanently	LED lit by central unit If the CU supports the acquisition of this (parameterised correctly)
1 LED per channel	red	lit permanently	ALARM LED lit by central unit If the CU calculates a threshold overflow on this channel (OPAL parameter setting)

VIII.10- Diagnostics of the Aoxxx analog output modules

Serigraphy	LED Colour	LED state	Signification
FLT	red	lit permanently	If general WDG Or if board internal power defect Or if no access via the CU software (if not parameterised)
1 LED per channel	green	lit permanently	LED lit by central unit If the CU supports the refresh of this channel

VIII.11- Diagnostics of the AIO320 analog input/output modules

Serigraphy	LED Colour	LED state	Signification
FLT	red	lit permanently	If general WDG Or if board internal power defect Or if no access via the CU software (if not parameterised)
1 LED per channel	orange	lit permanently	LED lit by central unit If a mechanism is connected at the input

Figure Table

FIGURE 1 : LT $\Leftrightarrow$ PC CONNECTION WIRING	1
FIGURE 2 : LT100 PRM MODE WIRING	3
FIGURE 3 : THE EXAMPLE NETWORK ARCHITECTURE.....	5
FIGURE 4 : PROCESSES APPLIED TO A DIGITAL INPUT	19
FIGURE 5 : PRINCIPLE OF DIGITAL INPUT FILTERING.....	20
FIGURE 6 : WIRING SECURITY INPUTS	21
FIGURE 7 : PROCESSES APPLIED TO DIGITAL OUTPUTS	22
FIGURE 8 : CONTINUOUS DIGITAL OUTPUT COMMAND.....	23
FIGURE 9 : FLASHING DIGITAL OUTPUT COMMAND.....	23
FIGURE 10 : MONOSTABLE DIGITAL OUTPUT COMMAND.....	24
FIGURE 11 BURST DIGITAL OUTPUT COMMAND	24
FIGURE 12 : FALLBACKS ON DIGITAL OUTPUTS.....	25
FIGURE 13 : PROCESSES APPLICABLE TO AN ANALOG INPUT	26
FIGURE 14 : SCALING OF AN ANALOG INPUT	27
FIGURE 15 : OVERFLOW OF UPPER/LOWER TYPE THRESHOLDS	28
FIGURE 16 : OVERFLOWS OF HYSTERESIS TYPE THRESHOLDS.....	28
FIGURE 17 : PROCESSES APPLICABLE TO AN ANALOG OUTPUT	29
FIGURE 18 : SCALING OF AN ANALOG OUTPUT.....	30
FIGURE 19 : PRODUCTION AND CONSUMPTION MECHANISM OF A FIP VARIABLE	40
FIGURE 20 : REFRESH STATUS MECHANISM ASSOCIATED WITH A FIP VARIABLE	41
FIGURE 21 : PROMPTITUDE STATUS MECHANISM ASSOCIATED WITH A FIP VARIABLE	41
FIGURE 22 : LT PROCESSES CYCLE	49
FIGURE 23 : DESCRIPTION OF THE EXECUTION PRINCIPLE OF THE <u>_MANDB</u> FUNCTION.....	58
FIGURE 24 : SYNTHESIS OF TIME FUNCTIONS	66